EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Enginyeria Informàtica. Pla 1997

Títol: Llibreria pel control estadístic de processos industrials per lots. Aplicació en una planta de depuració d'aigües

Document: Memòria

Alumne: Llorenç Burgas Nadal

Director/Tutor: Joaquim Meléndez, Joan Colomer
Departament: Enginyeria Elèctrica, Electrònica i Automàtica
Àrea: ATC

Convocatòria (mes/any): Febrer 2013

Agraïments

Primer de tot m'agradaria agrair a tots els integrants del Grup eXiT en el qual s'ha dut a terme aquest projecte la seva ajuda, en especial m'agradaria agrair a en Xavier Berjaga la seva paciència i consells, sense els quals no m'hauria estat possible acabar aquest projecte.

També m'agradaria agrair als tutors del PFC Dr. Joaquim Meléndez i Dr. Joan Colomer les seves correccions.

Índex

1 Introducció

Aquest capítol introductori serveix per situar al lector d'aquesta memòria de PFC en el marc on s'ha fet el treball, a més també introduirem els antecedents, motivacions, objectius a assolir i la planificació feta per dur-ho a terme.

1.1 Antecedents i motivacions

En el grup d'investigació d'Enginyeria de Control i Sistemes Intel·ligents (eXiT) es fa recerca en sistemes de monitorització i suport a la presa de decisions que integren solucions basades en l'explotació de dades històriques dels processos a supervisar. Solucions basades en la integració de tècniques com l'Anàlisi de Components Principals (*Principal Component Analysis*, PCA) i el raonament Basat en Casos (*Case Based Reasoning*, CBR) han donat bons resultats de recerca per la supervisió de processos industrials com la injecció de motlles, estacions de depuració d'aigües entre altres. Això ha motivat la creació d'una Toolbox de Matlab que permeti l'ús unificat dels algorismes desenvolupats en un únic entorn.

Fins el moment les eines desenvolupades utilitzen una estructura de dades bastant similar per enregistrar les dades del procés però en haver estat desenvolupades per diferents persones les crides i les estructures utilitzades com output són diferents. Un objectiu serà per tant unificar la forma en que es fan les crides i les estructures de dades que s'utilitzen.

També existeix una aplicació desenvolupada de forma conjunta amb el grup LEQUIA per a l'anàlisi de processos en línia. Aquesta aplicació està desenvolupada en LabView i utilitza Matlab amb la PLS Toolbox per la creació dels models. En algunes de les aplicacions existents es necessita tenir la PLS

Toolbox instal·lada ja que s'utilitzen crides a funcions d'aquesta Toolbox.

Aconseguir unificar tot el codi existent al grup en una única Toolbox, auto continguda (que no necessiti d'altres Toolboxes de Matlab), permetrà que el codi resultant sigui fàcilment reutilitzable d'aquí en endavant. També s'aconseguirà que les futures ampliacions tinguin un punt de partida estable i fàcil de modificar. El fet que el codi sigui auto-contingut (que no depengui de Toolboxes externes al grup) també facilitarà la reutilització i exportació dels codis del grup al no dependre de llicències d'altres Toolboxes.

1.2 Objectius del projecte

Com ja s'ha mencionat en la introducció aquest projecte de final de carrera es planteja generar un conjunt d'eines (Toolbox) per Matlab que permetin l'anàlisi i monitoratge estadístic d'un procés per lots, i que proporcionin les següents funcionalitats:

- Creació del model estadístic per al monitoratge del procés. Aquest model es realitza amb dades històriques adquirides en condicions normals (sense falles) i es realitza fora de línia (sense connexió directe amb el procés). Un cop s'ha obtingut aquest model, s'utilitzarà en línia per determinar les condicions en que es troba el procés a mesura que es reben noves dades.

- Disseny i implementació d'un procediment per a la diagnosi de la falla basat en criteris estadístics multivariants i raonament basat en casos. Igual que succeïa amb la creació del model, aquesta tasca primerament es realitza fora de línia (es determina la millor configuració de la metodologia per tal de diferenciar entre els diferents tipus de falla que es volen analitzar), i un cop establerta la configuració idònia, s'aplicarà a les noves observacions que es vagin rebent del procés.

- Dissenyar i implementar un conjunt de funcions per a la visualització de resultats, que facilitaran a l'operari la comprensió i identificació de l'estat del procés. La visualització dels resultats es pot realitzar tant en línia (quan s'aplica a les noves mostres que van arribant), així com fora de línia (per exemple, per analitzar el conjunt de dades que s'ha utilitzat per a la generació del model del procés).

-Inclusió de funcionalitats de preprocessat a les observacions que millorin la qualitat del model MPCA (detecció d'outliers, subdivisió del procés en fases, etc.).

-Creació d'un mòdul d'entrada/sortida de dades que permeti treballar amb diversos formats d'entrada (fitxers amb diferents formats o extensions, bases de dades...).

-Desenvolupar l'entorn de monitoratge en línia, amb capacitat d'accedir a dades del procés directament, així com la interfície d'usuari per accedir a les funcions de la Toolbox desenvolupada.

-Testejar i validar la Toolbox.

A mode d'exemple, i també com a validació de la Toolbox en un entorn real, s'implementarà una aplicació capaç de detectar el nivell de granularitat d'un procés de depuració d'aigües.

1.3 Planificació

Aquest projecte podem entendre'l com 3 etapes diferenciades. Una primera etapa on es crearan un conjunt de llibreries i funcions per tal de poder aplicar PCA i CBR sobre dades genèriques i testejar-ne el funcionament. La segona etapa de desenvolupament consisteix de dotar aquestes llibreries de més eines com la possibilitat de crear mes gràfics de control i un seguit de interfícies gràfiques per tal de facilitar les coses als usuaris. La darrera etapa d'aquest projecte consisteix en aplicar la Toolbox creada en un exemple d'utilització real.

El fet de tenir parts com la creació de interfícies gràfiques on cal molta interacció amb els usuaris ens va fer decantar per una metodologia àgil, per permetre la inclusió de modificacions fàcilment. Concretament la metodologia utilitzada ha estat l'Extreme Programming .

La creació de les llibreries, testeig d'aquestes ha estat fàcil de planificar i no ens hem desviat massa de dels terminis previstos. En canvi a l'hora de planificar la generació de la documentació hem fet curt ja que ha resultat ser una tasca més costosa i feixuga del que havíem previst inicialment.

Tot i utilitzar una metodologia àgil planificar la creació de les interfícies ha estat una tasca pràcticament impossible perquè en aquesta part s'han anat fent diverses versions i posteriorment progressives modificacions segons les indicacions dels usuaris (en el nostre cas tutors del PFC i companys del grup).

A la Figura 1 hi tenim la planificació, el punt on hem fet més curt ha estat la redacció de la documentació, tot i que s'ha anat fent progressivament a mesura que el projecte avançava. El fet de fer progressives modificacions en la implementació de les interfícies no ha afectat, perquè havíem previst que passaria i havíem estat generosos en assignar temps a les tasques de desenvolupament.

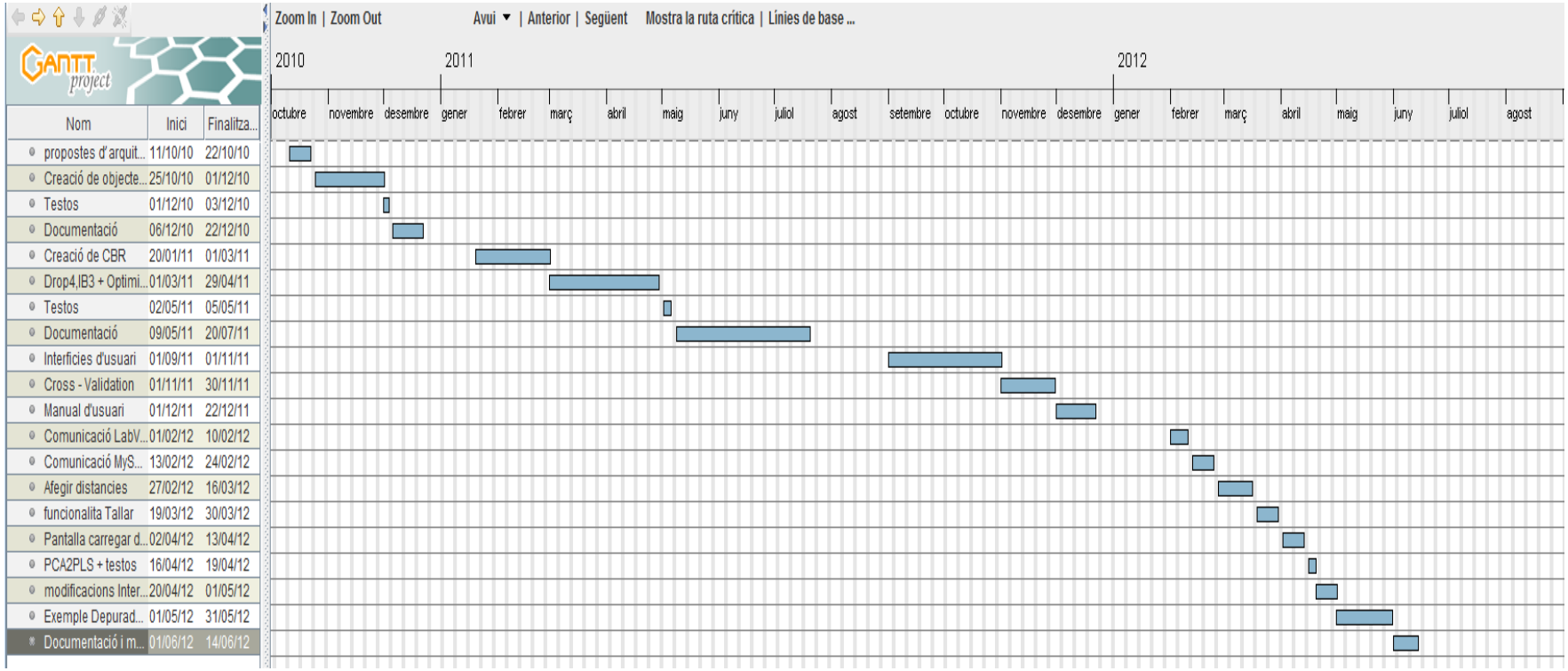

Zoom In | Zoom Out
Avui | Anterior | Següent
Mostra la ruta crítica | Línies de base ...
2010
2011
2012
octubre
novembre
desembre
gener
febrer
març
abril
maig
juny
juliol
agost
setembre
octubre
novembre
desembre
gener
febrer
març
abril
maig
juny
juliol
agost
Nom
Inici
Finalitza...
propostes d'arquit... 11/10/10 22/10/10
Creació de objecte... 25/10/10 01/12/10
Testos 01/12/10 03/12/10
Documentació 06/12/10 22/12/10
Creació de CBR 20/01/11 01/03/11
Drop4,IB3 + Optimi... 01/03/11 29/04/11
Testos 02/05/11 05/05/11
Documentació 09/05/11 20/07/11
Interficies d'usuari 01/09/11 01/11/11
Cross - Validation 01/11/11 30/11/11
Manual d'usuari 01/12/11 22/12/11
Comunicació LabV... 01/02/12 10/02/12
Comunicació MyS... 13/02/12 24/02/12
Afegir distancies 27/02/12 16/03/12
funcionalita Tallar 19/03/12 30/03/12
Pantalla carregar d... 02/04/12 13/04/12
PCA2PLS + testos 16/04/12 19/04/12
modificacions Inter... 20/04/12 01/05/12
Exemple Depurad... 01/05/12 31/05/12
Documentació i m... 01/06/12 14/06/12

Figura 1

El temps real aproximat dedicat a cada tasca del projecte, en ordre cronològic és el següent:

(3 setmanes) Plantejament i estudi de les diverses arquitectures possibles i aprenentatge bàsic sobre PCA

(3 mesos) Creació de objecte DATA i PCA + Testos per detectar errors + documentació corresponent

(2 mesos) Aprenentatge basic sobre CBR i Creació de l'objecte CBR

(2 mesos) Drop4, ib3, ib2, ibUdG + optimització a tota la llibreria

(1 més) Documentació corresponent a CBR

(2 setmanes) Cross-validation + testos

(2 mesos) interfícies usuari

(1 més) Documentació (Manual usuari)

(2 setmanes) Afegir noves distàncies, nous mètodes de reuse i optimitzacions

(2 setmanes) Funcionalitat Tallar, Permetre carregar dades des del WS als objectes

(1 setmana) PCA2PLS + fer exemple d'utilització Ethcal

(3 setmanes) Creació de la pantalla veure veïns

(2 setmanes) Fusió de diverses pantalles de PCA en 1 sola per millorar la facilitat d'us

(2 setmanes) Afegir a les pantalles informacions sobre el model, diagrames mostrant el % de casos de cada classe en temps real.

(4 setmanes) Crear rutines Lectura de GRASTAC, crear els models i provar configuracions + ajudar amb l'article

2 Coneixements previs

Aquest capítol és introductori, aquí s'intentaran introduir de la manera més assequible possible tots els conceptes necessaris per a la comprensió de la feina feta durant la realització d'aquest projecte de final de carrera.

2.1 Control de processos per lots

Un procés industrial per lots és aquell procés industrial amb el que es vol obtenir un producte acabat, on s'especifiquen una relació de materials a utilitzar (matèries primeres) i la forma com es combinaran aquests al llarg del temps i de quina manera, així com la condició que determina quan s'ha obtingut el producte final (un temps màxim de processat, l'assoliment d'un grau de "maduresa" del producte, etc.). Un cop estan fixades aquestes condicions, el procés s'anirà repetint de forma cíclica i on s'espera que la qualitat del producte sigui sempre la mateixa.

Malauradament, i com a conseqüència de la variació de la qualitat de les matèries primeres, el desgast dels components mecànics que conformen el procés, o simplement d'errors, al llarg del procés de producció, es donarà el cas que hi ha productes, que tot i que parteixen de la mateixa configuració, presenten variacions en les especificacions finals d'aquest, i en el pitjor dels casos, s'obtindran productes finals que no compleixen les especificacions de qualitat.

Per tant, és obvia la necessitat de monitorar el procés al llarg del temps per detectar qualsevol desviació no desitjada del procés, que desencadenaria en un producte fora de les especificacions de qualitat, quan abans millor (detecció de falles), que permetrà identificar la causa de l'anomalia (diagnosi de la falla), i que

en un últim pas, permetrà la correcció del procés.

El principal inconvenient a l'hora de monitorar aquest tipus de processos és que les condicions de control varien en funció del temps, i per tant, s'hauria de definir un "model" de comportament mentre les condicions del procés es mantenen. Al mateix temps, l'existència de registres històrics permet la utilització de mètodes estadístics que permeten identificar com varien aquestes relacions al llarg del temps de les diferents variables que es mesuren del procés.

2.2 PCA

PCA són les sigles de *Principal Component Analysis*. PCA és una tècnica que aprofita la correlació de les dades per reduir-ne la dimensió. Per fer-ho, PCA trobarà les combinacions de variables o factors d'aquestes variables que descriguin el màxim possible les variables inicials. Així doncs PCA es concentra a explicar la relació existent entre les variàncies i les covariàncies de les variables originals, a través de un seguit de transformacions lineals. Aquesta tècnica és utilitzada com a tècnica de compressió i interpretació de dades.

Els processos en els quals hi ha involucrats un gran nombre de variables poden ser monitoritzats utilitzant aquesta tècnica. Les observacions fetes durant el funcionament del procés en condicions de treball normals s'utilitzen per construir un model de dades. Posteriorment aquest model s'utilitza per analitzar el comportament del procés comprovant les noves dades projectant-les sobre el model.

2.2.1 Procediment

PCA és un algorisme que treballa sobre matrius bidimensionals. Segons PCA el conjunt de dades inicials (X) s'entén com una matriu on les files són les observacions i les columnes les diferents variables. Les variables hauran d'estar centrades (mitja 0) i estandarditzades (variància unitària). Un cop garantits aquests requisits podrem aplicar PCA.

Segons PCA un conjunt de dades X pot ser expressat com un conjunt de **r** noves variables (Components Principals), assumint una matriu d'error. La formulació matemàtica bàsica de PCA és la que trobarem a l'Equació 1.

$$X = TP' + E$$

Equació 1

A l'Equació 1 $T\ i\ P$ són el que s'anomenen matrius de *scores* i *loadings* respectivament. I P' és la matriu transposada de P. En aplicar PCA s'assumeix que els loadings associats a als valors propis de la matriu de correlació de X. Els valors propis representen les millors direccions per representar les dades segons el criteri de màxima variància.

La matriu de covariàncies S de les dades X es calcula tal com indica la Equació 2 on X' es X trasposta i n el nombre d'observacions de X.

$$\boldsymbol{S = \frac{1}{n-1} X'X}$$

Equació 2

Els vectors propis compleixen Equació 3 on V és una matriu ortogonal on les columnes seran els *loadings* i Λ és una matriu diagonal que conte els valors propis ordenats en ordre descendent.

$$S = V\Lambda V'$$

Equació 3

Si es compleixen totes les condicions esmentades al paràgraf anterior, representant només unes quantes components principals es possible representar la major part de la variància del conjunt original de dades. La informació no retinguda és representada per E, normalment associada a sorolls i variacions poc significants. Les primeres **r** components principals construeixen un nou espai amb una menor dimensionalitat que l'espai original de variables del procés. Seleccionar el nombre de components principals adient a cada problema no és fàcil ja que els resultats en depenen directament.

2.2.2 Projecció

La característica més important de PCA és la reducció del nombre de variables del problema. Aquesta reducció s'aconsegueix seleccionant les primeres **r** columnes de la matriu de *loadings* per construir la matriu P. $P \in R^{m \times r}$ associada amb els primers **r** components principals. Les projeccions de les observacions X en l'espai de menor dimensió s'inclouran en la matriu d'*scores* normalment anomenada T, que es calcula com s'indica a l'Equació 4.

$$T = XP$$

Equació 4

El pas invers (de T recuperar les dades en l'espai amb m variables originals) és simplement l'operació inversa, tal com s'indica a l'Equació 5.

$$\hat{X} = TP'$$

Equació 5

La diferencia entre X i $\hat{X}$ com s'indica en l'Equació 6 és la matriu d'errors E.

$$E = X - \hat{X}$$

Equació 6

Aquesta matriu conté un vector per cada observació, aquest vector és ortogonal a les components principals i resumeix la variància no capturada per les **r** components seleccionades en el nou espai de coordenades. Les components principals estan ordenades de manera que la primera component principal explica el màxim possible de la variabilitat de les dades. Cada component principal següent explica el màxim possible de la variabilitat restant en les dades originals.

2.3 Mètodes de selecció de components principals

PCA és molt sensible al nombre de components principals escollit. Per tal d'escollir el nombre de components adient per al problema existeixen diversos mètodes. A continuació explicarem els que hem inclòs a la llibreria.

2.3.1 Kaiser-Guttman

Consisteix en quedar-se només aquelles components principals que tinguin valors propis superiors a 1. Més intuïtivament consisteix en quedar-se totes les components que aportin com a mínim tanta informació com una de les variables inicials.

Aquest mètode tendeix a agafar menys components principals de les realment necessàries.

2.3.2 Scree Test

Aquest mètode utilitza el fet que els valors propis, tenen una forma característica quan els representem en una gràfica (una corba decreixent que presenta un descens pronunciat al principi i una asímptota horitzontal a 0). Segons aquest mètode el nombre ideal de components principals és troba el punt on la gràfica dels valors propis presenta el colze.
Aquest mètode tendeix a agafar més components principals del necessari.

2.3.3 % de variància explicada

Aquest mètode de selecció de components principals consisteix en quedar-se tantes components principals com faci falta per tal que el nou conjunt de variables representi un cert % de la variància de les variables originals. El principal problema d'aquest mètode és la dificultat d'escollir el % de variància necessari. Depenent del problema el % de variància necessari varia i és difícil de predir a priori.

2.3.4 VRE

VRE són les sigles de *Variance of the Reconstruction Error*, i es un mètode de selecció del nombre components principals. Quan treballem amb PCA la informació pot veure's reflectida a les components principals o bé a la matriu d'errors, normalment quantes més components principals agafem per fer el model menys informació hi haurà a la matriu d'errors. Aquesta tècnica a l'hora d'escollir el nombre de components principals intenta assegurar que tant a les components principals com a la matriu d'error quedi com a mínim informació suficient per poder reconstruir les dades.

2.4 Estadístics més comuns per PCA

Per tal de complementar PCA amb gràfics de control es defineixen normalment per processos multivariants els estadístics Q , $T^{2,}$ i es calculen les contribucions de les variables originals a cadascun d'aquests.

2.4.1 T^2

Es calcula per cada observació sumant els **r** components principals al quadrat i dividint per les variàncies. Es pot entendre com la distància (Mahalanobis) entre una observació i el centroide del model. Valors elevats de T^2 indica que estem davant comportaments atípics. L'expressió analítica d'aquest estadístic és la que es mostra a l'Equació 7.

$$T^2 = \sum_{j=1}^{r} \frac{t_j^2}{\sigma_j^2}$$

Equació 7

2.4.2 Q

Aquest estadístic està relacionat amb la matriu d'errors, sensible a canvis de l'estructura del procés. en condicions normals Q presenta valors baixos, en condicions anòmales Q presentarà valors elevats. L'expressió analítica d'aquest estadístic és la que es mostra a l'Equació 8.

$$Q = \sum_{j=1}^{m} (x_j - \hat{x}_j)^2$$

Equació 8

2.4.3 Anàlisi de Contribucions

La intenció d'aquest estudi és detectar quina de les variables del procés estan relacionades amb una falla. Depenent de l'estadístic que hagi detectat la falla (Q o T^2) les contribucions es calcularan de manera diferent donant dos tipus de contribucions les contribucions de Q i les contribucions de T^2.

2.4.3.1 Contribucions de T^2

La contribució de cada una de les j variables d'un procés es defineix tal com indica l'Equació 9.

$$ct_j = \sum_i p_{ij} x_j \frac{t_i}{\sqrt{\lambda_i}}$$

Equació 9

A l'Equació 9 t_i i λ_i representen el valor i la variança dels i-èssim *score* de la j-èsima variable del procés i p_{ij} és l'element (i,j)-èsim se la matriu P (per mes detalls sobre la matriu P veure apartat 2.2.2).

2.4.3.2 Contribucions de Q

Les contribucions de Q en indiquen quina variable és la que provoca la falla. I es calcula tal com mostra la Equació 10.

$$cq_j = (x_j - \hat{x}_j)^2$$

Equació 10

2.4.4 Signatures

Com que les contribucions de Q i T^2 presenten una distribució normal podem considerar que tots els valors que superin la mitja +- tres desviacions estàndard són valors anòmals. Podem buscar les signatures agrupant per fases o

simplement indicant si falla o no en un determinat instant.

$$Sj = c_j - mean(c) \mp 3 * std(c)$$

Equació 11

A la Equació 11 hi tenim la fórmula de com es calculen les signatures, on Sj són les signatures d'un determinat cas, c_j són les contribucions del cas Sj i $mean(c)$ i $std(c)$ són la mitja i la desviació estàndard de totes les contribucions del procés.

2.5 PCA per processos per lots (MPCA)

PCA per definició es pot aplicar sobre matrius bidimensionals (observacions x variables). Els processos per batchs o lots són normalment processos representats per series d'observacions al llarg del temps. Conseqüentment, la matriu per representar aquest tipus de processos és tridimensional (observacions x variables x temps). Aquesta dimensió extra ens obligarà a fer un preprocessat de les dades per tal de poder aplicar PCA (Unfolding i escalat). Nosaltres treballarem amb una matriu tridimensional estructurada com la de la Figura 2. On les dimensions equivaldran a:

Dimensió1 → Temps (Observacions dins un mateix lot)

Dimensió2 → Variables

Dimensió3 → Lots

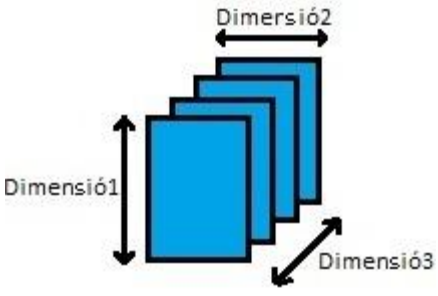


Figura 2

2.5.1 Unfolding

Per realitzar l'unfolding o desdoblament d'una matriu tridimensional se'ns presenten dues possibilitats batch-wise i variable-wise.

El procediment de desdoblar consisteix, en el cas de batch-wise, en mantenir com a files els lots i desdoblar com a columnes les variables al llarg del temps. Les dades de la Figura 2 quedarien després de desdoblar en batch-wise posicionades tal com es mostra a la Figura 3. On les dimensions equivaldran a:

Dimensió1 → Lots
Dimensió2 → Variables x Temps

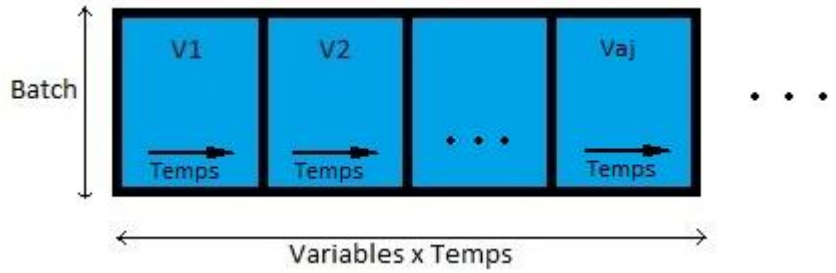


Figura 3

En cas d'utilitzar un desdoblament en variable-wise. Observarem que, a les columnes es mantindran les variables, mentre que en les files de la nova matriu bidimensional hi trobarem els diversos lots al llarg del temps. Les dades de la Figura 2 quedarien després de desdoblar en variable-wise posicionades tal com es mostra a la Figura 4. On les dimensions equivaldran a:
Dimensió1 → Batch x Temps
Dimensió2 → Variables

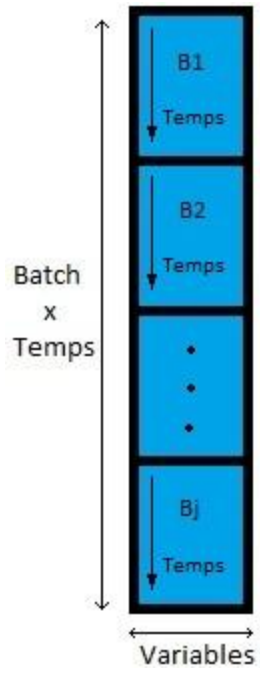


Figura 4

2.5.2 Escalat

Les dades per aplicar PCA han d'estar centrades i estandarditzades. En realitzar un desdoblat batch-wise o variable-wise cal tenir en compte que no totes les variables tenen el mateix rang de valors al calcular la mitjana i la desviació estàndard. Per evitar que algunes variables prenguin més importància que la resta degut a la seva tipologia es defineixen un seguit de funcions d'escalat.

Continuous scaling (CS) – Assumeix que les variables tenen la mateixa distribució. Es calcularà una mitja i una desviació estàndard per cada variable original.

Group Scaling (GS) – Treu la trajectòria mitja de les variables. La mitjana es calcula per cada instant de cada una de les variables. La desviació estàndard es calcula per cada variable original.

Auto Scaling (AS) – Per quan la variabilitat canvia al llarg del procés. La mitjana i la desviació estàndard es calcula per cada instant de cada una de les variables.

2.6 CBR

CBR (*Case Based Reasoning*) es pot descriure de moltes maneres, la més coneguda (representada gràficament a la Figura 5) és descriure el CBR com un procés cíclic de quatre fases que giren al voltant de l'experiència del sistema, l'experiència del sistema s'emmagatzema en una estructura anomenada base de casos. Davant d'un nou problema, el sistema realitza els passos següents:

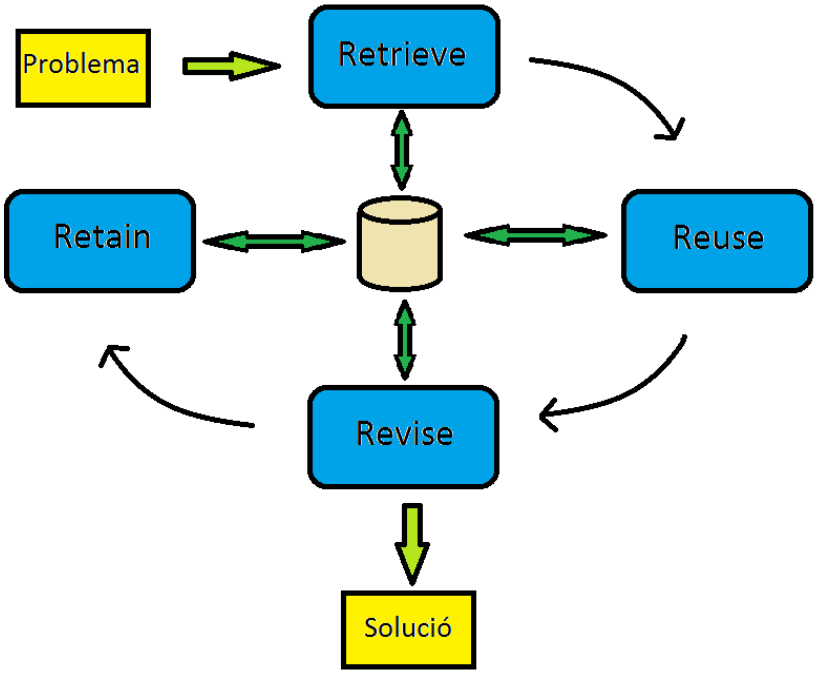


Figura 5

Fase de Retrieve: Es busquen els casos més semblants de la base de casos respecte el problema nou. Dins d'aquest procés juguen un paper clau l'organització de la memòria de casos i la manera de calcular les distàncies entre els casos (més detalls a l'apartat 2.6.1). El primer aspecte és important perquè

condiciona la informació que es consulta, i el segon és important perquè defineix el grau de semblança entre el problema nou i els emmagatzemats a la memòria. Aquesta fase és molt important ja que la resta de fases depenen dels seus resultats.

Fase de Reuse: Es proposa una solució a partir dels casos recuperats a la fase anterior. L'adaptació pot ser molt simple (assignar la mateixa resolució que el/s cas/os recuperat/s mètode del veí més proper) o tant complexa com es desitgi.

Fase de Revise: Es valida si la solució proposada és correcta a través de la figura de l'expert. No obstant, l'expert sovint no pot afirmar que la resolució sigui certa al 100%, només pot donar la seva opinió respecte si la resposta proposada té sentit i no és un disbarat. Això és degut a que si sabés la resposta, no caldria un sistema CBR que li donés. En alguns casos el paper de l'expert pot automatitzar-se amb l'ajuda de regles o heurístiques específiques al domini.

Fase de Retain: És la fase on l'algorisme decidirà quins casos cal guardar. Cal definir quins criteris han d'acomplir-se per tal d'incorporar el nou coneixement. L'aplicació d'una política incorrecte pot empitjorar la qualitat de la base de casos en tasques de raonament, impossibilitant la resolució dels casos nous. Per tant, cal assignar una política segons el domini que s'està estudiant, la qual mantingui dins de les possibilitats una memòria reduïda, compacte i representativa. (més detalls a l'apartat 2.6.2)

De la mateixa manera que en qualsevol altre sistema d'aprenentatge, el CBR requereix d'una fase prèvia per preparar i processar les dades. A partir de la definició del problema a resoldre, cal obtenir i quantificar les característiques més rellevants que descriuen el problema. Sovint aquestes valoracions no seran objectives ja que dependran de la percepció de l'expert, o de la fiabilitat de la mesura realitzada. A més a més, un cas està descrit per molts tipus de característiques i els seus valors tenen sovint soroll o valors desconeguts. Per

tant, tots aquests aspectes s'han de tenir en compte a l'hora de treballar.

En el nostre cas utilitzarem un CBR basat en PCA. És a dir els casos estaran representats per components principals i els seus estadístics (Q, T^2).

2.6.1 Distàncies CBR

CBR és un mètode on els resultats depenen sovint de la utilització d'una distància adient per al problema. En implementar la Toolbox hem hagut de parar atenció en aquest punt i dotar a l'usuari de les distancies més típiques i d'altres que hem cregut que podrien ser útils en segons quins problemes. A continuació descriurem els fonaments teòrics darrera de cada una de les distàncies implementades.

2.6.1.1 Distància sobre les components principals

Aquest criteri calcula la distància euclidiana entre els casos. En aquest criteri, el vector de característiques són les projeccions sobre cadascuna de les components principals que s'han retingut, normalitzades pel seu valor propi associat (λi), emprant la següent expressió:

$$d_t(c_a, c_b) = \sqrt{\sum_{i=1}^{r} \frac{(t_{c_a,i} - t_{c_b,i})^2}{\lambda i}}$$

Equació 12

On **r** és el nombre de components principals retingudes pel model PCA, t_{c_a} , i és la projecció del vector de característiques del cas c_a sobre la i-èssima component principal; t_{c_b} , és la projecció del vector de característiques del cas c_b sobre la i-èssima component principal; i λi és el valor propi associat a la i-èssima component principal. Finalment la Figura 6 mostra la projecció de casos

sobre les 3 primeres components principals d'un model PCA i com es calcularia la distància entre els Z casos c_a i c_b.

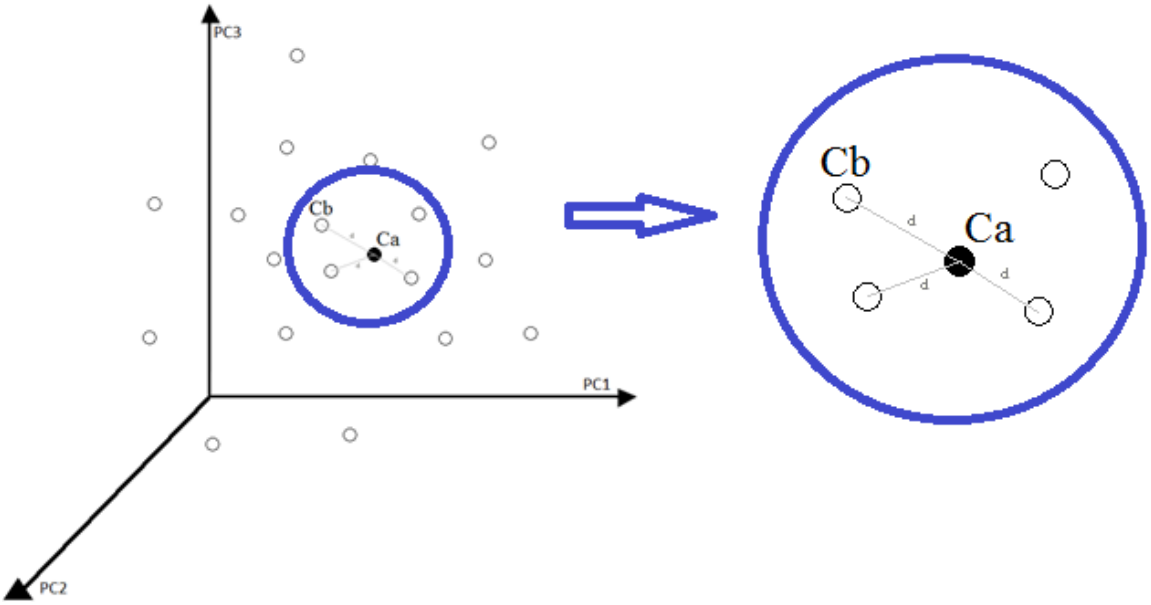


Figura 6: Distància per Components Principals

2.6.1.2 Distància sobre T²

Aquest criteri es basa en comparar els valors de T^2 entre 2 casos diferents (c_a i c_b) de la següent forma:

$$d_t(c_a, c_b) = |T^2{}_{c_a} - T^2{}_{c_b}|$$

Equació 13

On $T^2{}_{c_a}$ és el valor de T^2 per al cas c_a i $T^2{}_{c_b}$ és el valor de T^2 per al cas c_b. La representació gràfica d'aquest càlcul es mostra en la Figura 7.

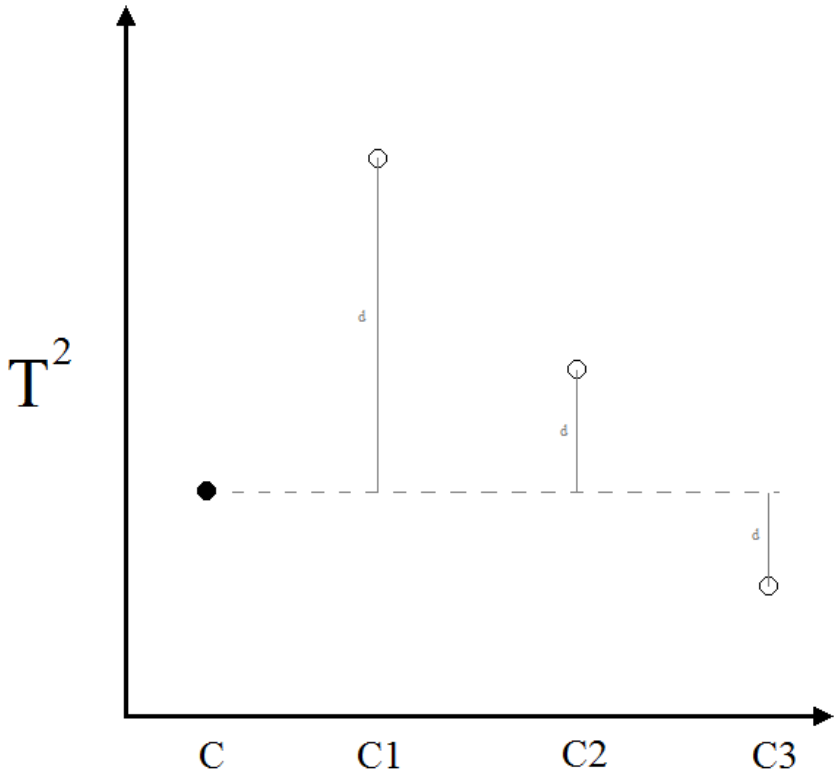


Figura 7: Distància per T^2

2.6.1.3 *Distància sobre Q*

Aquest criteri es basa en comparar els valors de Q entre 2 casos diferents (c_a i c_b) de la següent forma:

$$d_Q(c_a, c_b) = | Q_{c_a} - Q_{c_b} |$$

Equació 14

On Q_{c_a} és el valor de Q per al cas c_a i Q_{c_b} és el valor de Q per al cas c_b. La representació gràfica d'aquest càlcul es mostra en la Figura 8.

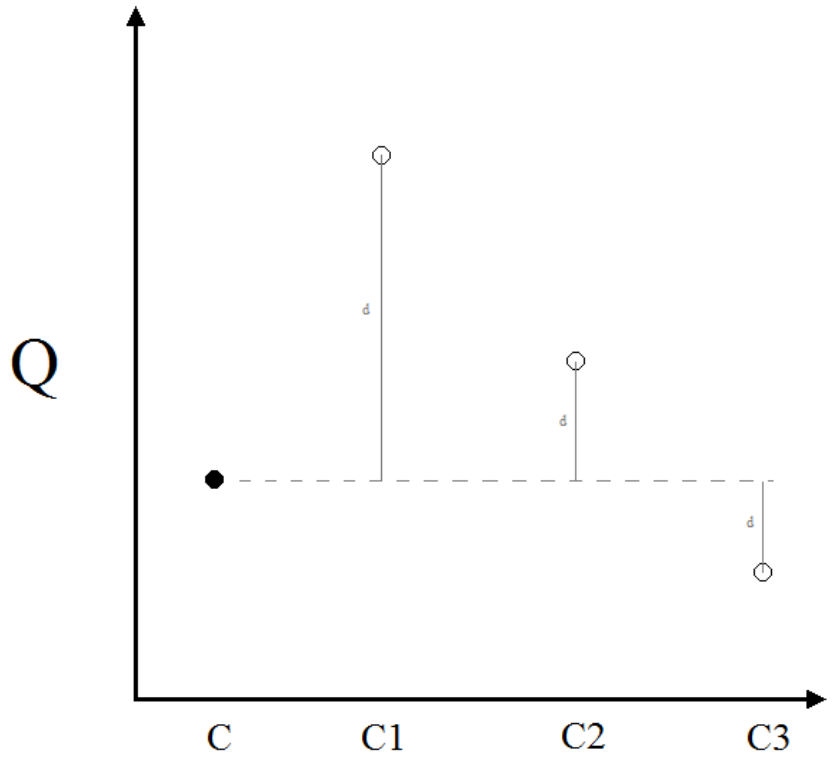


Figura 8 : Distància per Q

2.6.1.4 Distància combinada sobre Q i T^2

Les distàncies combinades ordenen els casos segons dos criteris de distància diferents. Per fer-ho primer ordenen els casos pel primer criteri (de menor a major distància). D'aquesta primera ordenació només retenim els primers "vk(1)" casos. Aquests "vk(1)" casos són reordenats de menor a major distància en funció del segon criteri, i dels quals només ens quedarem amb els "vk(2)" primers casos. El criteri de Q i T^2 primer ordena els casos segons el criteri de distància de Q (Equació 14) i un cop filtrats els "vk(1)" casos més propers els reordena segons el criteri de distància de T^2 (Equació 13). La representació gràfica d'aquests criteri es mostra en la Figura 9, on el punt negre és el cas sobre el què volem trobar els "vk" veïns més propers.

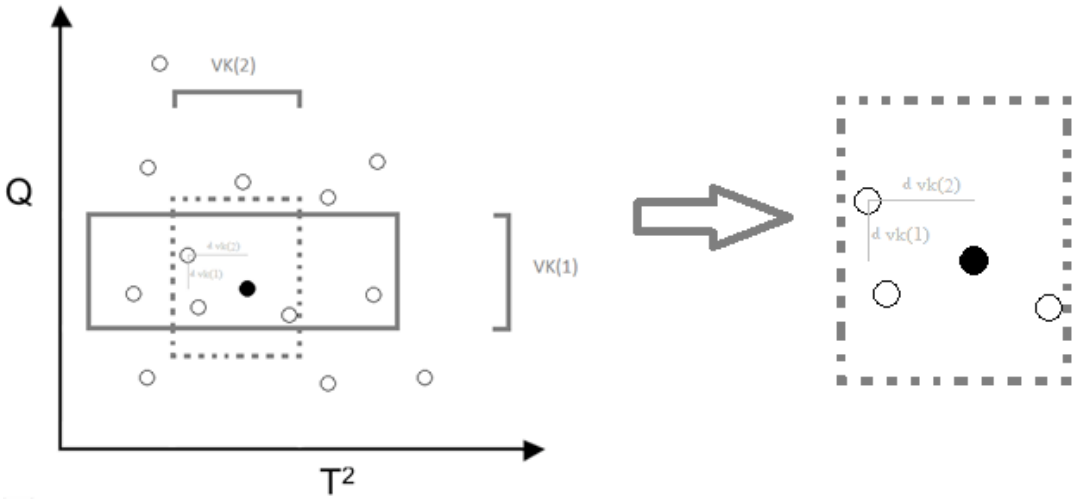


Figura 9 : Distància combinada per Q i T^2

2.6.1.5 Distància combinada sobre Q i les principals components

Pel cas de distància combinada de Q i les Components Principals, els casos primer s'ordenen pel criteri de distància de Q (Equació 14) i després, els "vk(1)" primers casos es reordenen segons el criteri de les Components Principals (Equació 12). La representació gràfica d'aquest criteri es mostra en la Figura 10.

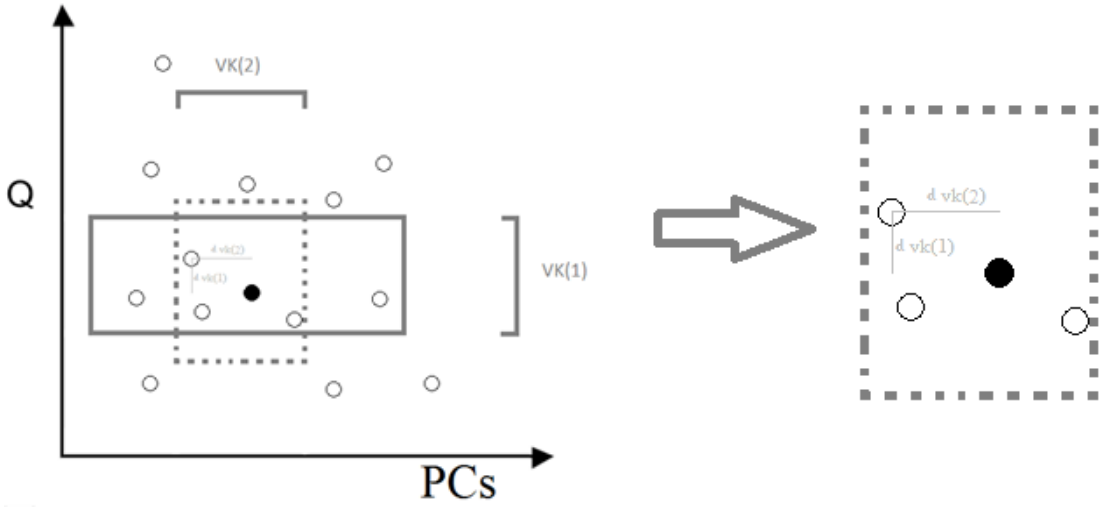


Figura 10 : Distància combinada per Q i les components principals

2.6.1.6 Distància combinada sobre T^2 i les principals components

En el cas de la distància combinada de T^2 i les components principals, els casos en primera instància s'ordenen segons els valors de T^2 (Equació 13), d'on només en retindrem els "vk(1)" casos més propers. Després aquests "vk(1)" casos es reordenen segons el criteri de les components principals (Equació 12), d'on només retenim els "vk(2)" primers casos. La representació gràfica d'aquest criteri és la que es mostra en la Figura 11.

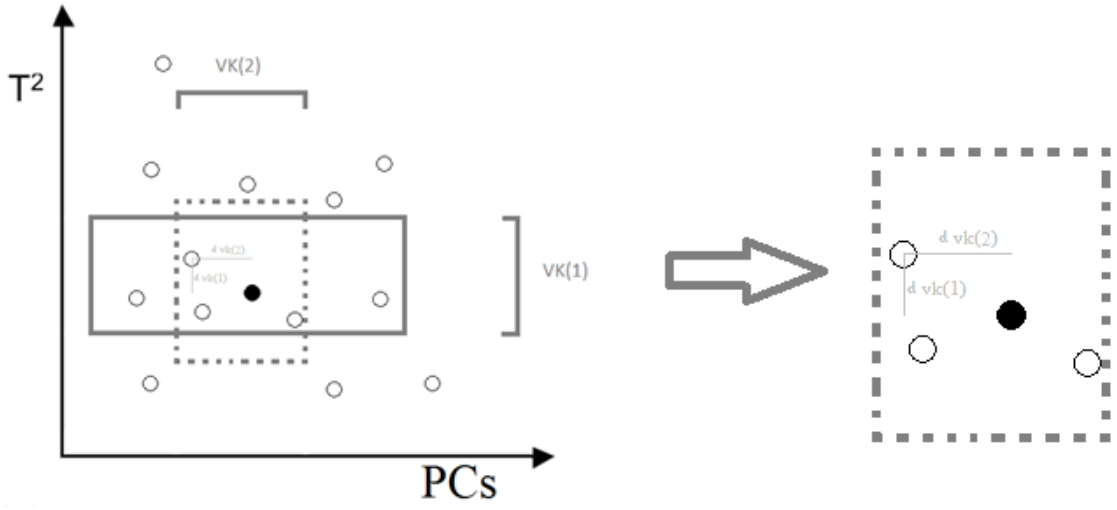


Figura 11 : Distància combinada per T^2 i les components principals

2.6.1.7 Distància sobre Contribucions de Q en distància Euclidiana

En el cas de la distància simple sobre les contribucions de Q , els casos s'ordenen segons el valor que dona la Equació 15. Podem observar que la fórmula és una distància euclidiana sobre els valors de les contribucions de Q dels casos "c_a" i "c_b".

$$d_{CQ}(c_a, c_b) = \sqrt{\sum_{i=1}^{r} (CQ_{c_a,i} - CQ_{c_b,i})^2}$$

Equació 15

2.6.1.8 Distància sobre Contribucions de T^2 en distància Euclidiana

En el cas de la distància simple sobre les contribucions de T^2, els casos s'ordenen segons el valor que dona la Equació 16. Podem observar que la fórmula és una distància euclidiana sobre els valors de les contribucions de T^2 dels casos "c_a" i "c_b".

$$d_{CT^2}(c_a, c_b) = \sqrt{\sum_{i=1}^{r} (CT^2_{c_a,i} - CT^2_{c_b,i})^2}$$

Equació 16

2.6.1.9 Distància sobre Contribucions de Q en distància Mahalanobis

En el cas de la distància simple sobre les contribucions de Q , els casos s'ordenen segons el valor que dona la Equació 17. Podem observar que la fórmula és una distància de Mahalanobis sobre els valors de les contribucions de Q dels casos "c_a" i "c_b".

$$d_{CQ}(c_a, c_b) = \sqrt{\sum_{i=1}^{r} \frac{(CQ_{c_a,i} - CQ_{c_b,i})^2}{\sigma^2}}$$

Equació 17

La utilitat d'utilitzar Mahalanobis radica en la forma de determinar la similitud entre dues variables aleatòries multidimensionals. Es diferencia de la distància euclidiana en que té en compte la correlació entre les variables aleatòries. Ponderant segons la seva variància: les variables amb menys variància tindran més importància que les de major variància. D'aquesta manera es pretén igualar la importància de les variables amb la distància.

2.6.1.10 Distància sobre Contribucions de T^2 en distància Mahalanobis

En el cas de la distància simple sobre les contribucions de T^2, els casos s'ordenen segons el valor que dona la Equació 18. Podem observar que la fórmula és una distància euclidiana sobre els valors de les contribucions de T^2 dels casos "c_a" i "c_b".

$$d_{CT^2}(c_a, c_b) = \sqrt{\frac{\sum_{i=1}^{r}(CT^2_{c_a,i} - CT^2_{c_b,i})^2}{\sigma^2}}$$

Equació 18

La utilitat d'utilitzar Mahalanobis radica en la forma de determinar la similitud entre dues variables aleatòries multidimensionals. Es diferencia de la distància euclidiana en que té en compte la correlació entre les variables aleatòries. Ponderant segons la seva variància: les variables amb menys variància tindran més importància que les de major variància. D'aquesta manera es pretén igualar la importància de les variables amb la distància.

2.6.1.11 Distància Euclidiana sobre Q i T2

A diferència de la distància combinada de Q i T^2, aquest criteri té en compte els dos valors al mateix nivell, calculant la distància entre casos com la distància euclidiana de la Equació 19.

$$d_{QiT^2}(c_a, c_b) = \sqrt{(Q_{c_a} - Q_{c_b})^2 + (T^2{}_{c_a} - T^2{}_{c_b})^2}$$

Equació 19

A la Equació 19 $T^2{}_{c_a}$ és el valor de T^2 per al cas c_a , $T^2{}_{c_b}$ és el valor de T^2 per al cas c_b , Q_{c_a} és el valor de Q per al cas c_a i Q_{c_b} és el valor de TQ per al cas c_b. La representació gràfica d'aquest calcul es mostra a la Figura 12.

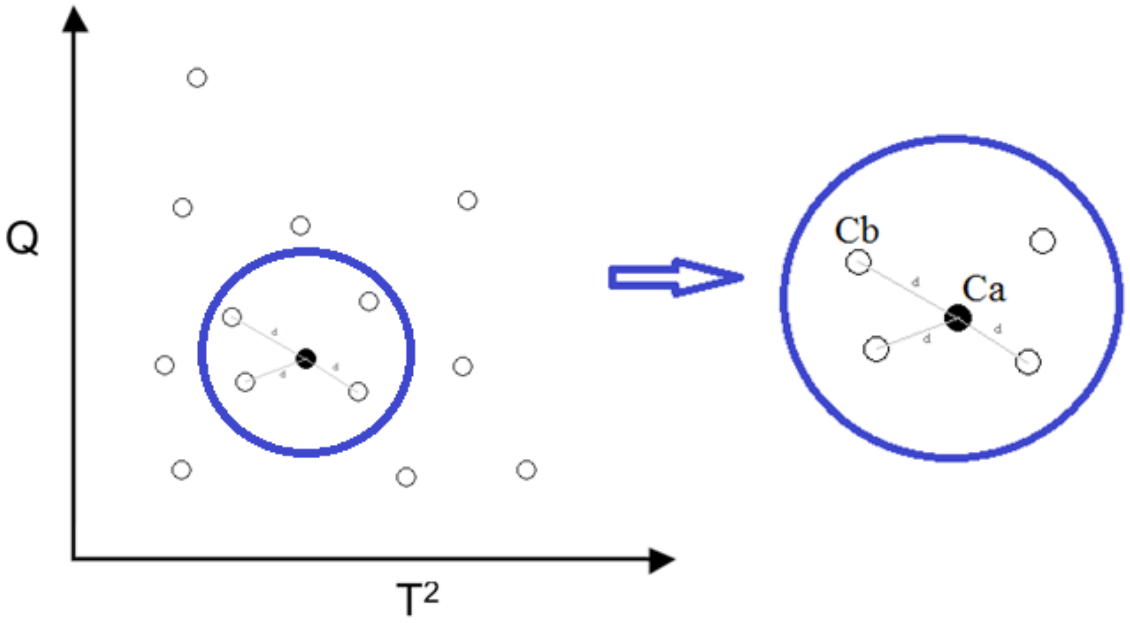


Figura 12 : Distància Euclidiana per Q i T^2

2.6.1.12 Distància sobre les signatures de Q

Tal com s'han definit les signatures a l'apartat introductori 2.4.4 podem utilitzar-les com a distancia en CBR. Per cada cas obtindrem un valor que ens indicarà si el lot és anòmal o no en aquell instant. El que farem al utilitzar aquesta distancia és fer que CBR busqui casos que on hagin fallat les mateixes variables i en els mateixos instants.

2.6.1.13 Distància sobre les signatures de T^2

Tal com s'han definit les signatures a l'apartat introductori 2.4.4 podem utilitzar-les com a distancia en CBR. Per cada cas obtindrem un valor que ens indicarà si el lot és anòmal o no en aquell instant. El que farem al utilitzar aquesta distancia és fer que CBR busqui casos que on la informació la aportin les mateixes variables i en els mateixos instants.

2.6.1.14 Distància sobre les signatures de Q agrupant per fase

Com la distancia explicada a 2.6.1.12 però treballant amb la suma dels resultats de cada una de les fases del procés. Agrupant per fases aconseguim que els resultats siguin més robustos a desfasaments.

2.6.1.15 Distància sobre les signatures de T^2 agrupant per fase

Com la distancia explicada a 2.6.1.13 però treballant amb la suma dels resultats de cada una de les fases del procés. Agrupant per fases aconseguim que els resultats siguin més robustos a desfasaments.

2.6.1.16 Distancia sobre les components principals sense normalització

Aquesta distancia calcula la distància euclidiana entre els casos. En aquest criter de distànciai, el vector de característiques són les projeccions sobre cadascuna de les components principals que s'han retingut, a diferencia de la distancia 2.6.1.1 sense normalitzar pel seu valor propi associat (λi). Tal com s'indica a Equació 20.

$$d_t(c_a, c_b) = \sqrt{\sum_{i=1}^{r} (t_{c_a,i} - t_{c_b,i})^2}$$

Equació 20

On **r** és el nombre de components principals retingudes pel model PCA, t_{c_a}, i és la projecció del vector de característiques del cas c_a sobre la i-èssima component principal i t_{c_b} és la projecció del vector de característiques del cas c_b sobre la i-èssima component principal. Finalment la Figura 6 mostra la projecció de casos sobre les 3 primeres components principals d'un model PCA i com es calcularia la distància entre els Z casos c_a i c_b.

2.6.1.17 Distància sobre les signatures de Q agrupant per fase banalitzant amb un llindar

Com la distancia explicada a 2.6.1.12 però treballant amb la suma dels resultats de cada una de les fases del procés. Agrupant per fases aconseguim que els resultats siguin més robustos a desfasaments. A diferencia de la distància 2.6.1.14 binaritzarem els valors de cada fase tractant la fase com un tot.

2.6.1.18 Distància sobre les signatures de T^2 agrupant per fase banalitzant amb un llindar

Com la distancia explicada a 2.6.1.13 però treballant amb la suma dels resultats de cada una de les fases del procés. Agrupant per fases aconseguim que els resultats siguin més robustos a desfasaments. A diferencia de la distància 2.6.1.15 binaritzarem els valors de cada fase tractant la fase com un tot.

2.6.2 Polítiques de manteniment de la base de casos

Sovint ens trobem que la base de casos en utilitzar CBR augmenta considerablement, per l'acció de la fase de Retain i probablement disminueix la precisió del model degut a l'overfitting.

En CBR un altre dels punts més importants és doncs el manteniment de la base de casos. CBR és una metodologia que va aprenent a mesura que es va fent servir, tot i això no sempre tots els casos ens són útils.

En una base de casos podem trobar-nos:

- Casos redundants
- Casos que provoquen equivocacions (Lamentables)

Existeixen diversos mètodes per eliminar aquests casos i així amb bases de casos més petites poder classificar de manera molt més ràpida els nous casos. Els algorismes més rellevants són IB2, IB3 i DROP4 tots ells han estat implementats en la nostra Toolbox, a continuació els introduirem breument:

- **IB2** - Mètode per netejar la base de casos proposat per David W. Aha . IB2 és un algorisme incremental (comença amb la base de casos buida i la va emplenant amb el casos que són necessaris), va provant de classificar casos un a un, en cas que es falli en classificar-lo el cas serà afegit a la base de casos definitiva, si per contra el cas es classifica bé no s'afegirà.
- **IB3** - Mètode per netejar la base de casos proposat per David W. Aha. Aquest mètode intenta definir les fronteres de les classes lo més allunyades possible. La frontera entre dues classes és la zona de l'espai de cerca del problema en que dues classes presenten una gran confluència.
- **DROP4** - Mètode per netejar la base de casos proposat per Wilson i Martinez. És un procés decremental , comença amb la base de casos completa i en treu els que no són necessaris. De manera simplificada consisteix en comprovar cas per cas si en treure'l es classifica millor la base de casos inicial o si pel contrari els resultats empitjoren. Si empitjoren el cas és necessari, si milloren el cas és redundant. A més a més incorpora un filtre per detectar els casos que anomena lamentables, casos envoltats d'elements d'altres classes.

2.7 Eines de Test

A continuació mostrarem algunes eines que permeten avaluar de manera objectiva el funcionament correcte d'un model per a tasques de classificació/predicció que serà les que aplicarem al fer la detecció i diagnòstic de falles.

2.7.1 n-Fold Cross Validation

En n-Fold Cross Validation, les dades disponibles es divideixen en **n** conjunts que contenen aproximadament el mateix nombre d'elements. Un cop dividides un dels conjunts es guarda per fer la validació del model format amb els altres **n-1** conjunts. Aquest procés es repeteix **n** cops un per cada conjunt de dades.
Utilitzant aquesta tècnica podrem demostrar que els resultats obtinguts són independents del conjunt de dades utilitzat i així podrem generalitzar els resultats obtinguts.

Com podem observar a la Figura 13 en aplicar n-Fold Cross Validation amb n=4, les Dades inicials són dividides en 4 parts, cada cicle una de les quatre parts es reserva com a test (dades pintades en vermell) i la resta és utilitzat per crear el model i entrenar-lo (dades pintades en blau). El model resultant a cada cicle després de l'entrenament corresponent s'utilitza per classificar les dades prèviament apartades com a test. Dels resultats de classificació dels casos de test és d'on s'extreuen els estadístics de classificació.

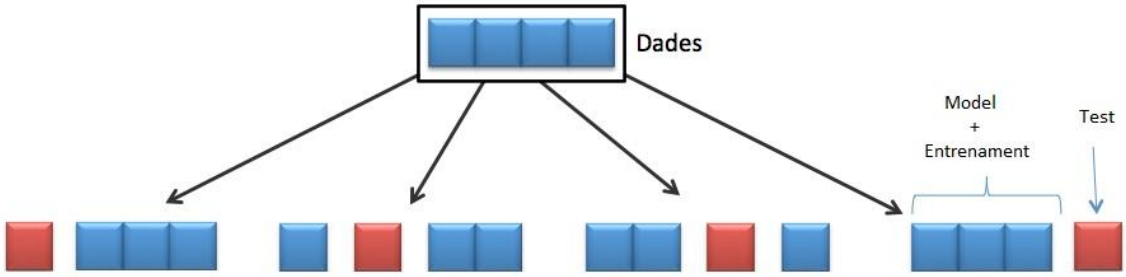


Figura 13

2.7.2 Matriu de confusió

És una eina molt útil per avaluar la classificació. Una matriu de confusió és una eina de visualització de resultats. Cada columna de la matriu representa el nombre de prediccions de cada classe d'un conjunt de dades prèviament etiquetades, mentre que cada fila representa les instàncies en la classe real. Una de les principals utilitats de les matrius de confusió és que faciliten veure si el sistema està confonent dues classes.

		Classe real	
		Classe 1	No Classe 1
Classe predita	Classe 1	TP	FP
	No Classe 1	FN	TN

TP i TN són instancies que el model ha classificat bé. FN i FP són instancies que el model no ha classificat bé i corresponen a falsos negatius (FN) i falsos positius (FP).

3 Disseny i Implementació

En aquest capítol hi trobarem els detalls de com s'ha implementat aquest projecte. Començarem presentant el codi que ja existia anteriorment a aquest PFC i com l'hem incorporat. Continuarem explicant els requisits necessaris per poder utilitzar la Toolbox i el perquè de cada un d'ells. Finalment exposarem com estan dissenyades les llibreries.

3.1 Codi existent

Per desenvolupar aquest PFC s'han aprofitat codis procedents de diverses fonts. A continuació descriurem quins són, què fan i d'on prové el codi. D'aquesta manera es deixarà clar quina part del codi del projecte és codi reaprofitat, codi millorat o adaptat i quina part és codi nou.
Les aportacions externes les subdividirem en 2 grups, codi procedent d'altres projectes del grup i codi lliure procedent de fonts externes al grup .

3.1.1 Codi procedent de integrants del grup eXiT

Com s'esmenta en la introducció, en començar el PFC se'ns van proporcionar diversos codis que treballen amb PCA i CBR, tots ells amb diverses funcionalitats i objectius. Com a punt de partida es va encarregar crear una llibreria que englobés totes aquestes funcionalitats, i que fos tan genèrica com fos possible.

El codi proporcionat esta format per 3 llibreries, independents entre elles. Cada una d'elles treballa sobre la seva pròpia estructura de dades mitjançant el conjunt de funcions que integra la llibreria.

Llibreria PCA Anàlisi de contribucions

Aquest codi es centra a dotar a PCA de gràfics i les eines necessàries per al càlcul de les contribucions de Q i T^2. Pel que fa a PCA aquest codi depèn de una Toolbox per Matlab anomenada PLS Toolbox.

Llibreria CBR

Codi que permet aplicar CBR sobre PCA. Per tal d'aplicar PCA aquest codi utilitza la PLS Toolbox. A CBR se li han definit 3 distàncies que son escollides automàticament segons la procedència de les dades:

- Components principals i Q – S'utilitza una distància euclidiana en 2 nivells.
- Contribucions –Distancia mahalanobis.
- Contribucions binaritzades (Descriptors)– s'utilitza la distancia de Hamming.

Llibreria MPCA CBR Library

Llibreria que defineix funcions que permeten aplicar PCA sobre matrius tridimensionals de dades, realitzant –ne l'estandardització i desdoblat escollint d'entre diverses opcions. També permet aplicar VRE d'entre d'altres mètodes de selecció de components i les utilitats de retrieve i reuse de CBR.

Un cop introduïdes les 3 llibreries farem un petit resum de les diverses tècniques que trobem implementades en aquestes 3 llibreries:

- PCA
    - Estandardització
    - Unfolding
    - Analisi de Contribucións
    - Exportar dades a Weka
    - Utilitzar VRE

- CBR
    - Retrieve
    - Reuse
    - Revise/retain mitjançant:
        - Drop4
        - IB3
- Cross validation
- Parser fitxers Grasstac

Com podem observar, caldrà crear una estructura que permeti integrar aquestes 3 llibreries, i adaptar els codis que ho necessitin perquè funcionin sobre dades genèriques.

3.1.2 Codi de lliure distribució

Per tal de desenvolupar la Toolbox hem utilitzat codis d'altres persones, extrets tots ells del portal File Exchange proporcionat pel propi Matlab perquè els usuaris intercanviïn els seus codis lliurement. Tots els codis que hem utilitzat són de lliure distribució.

Les funcions externes que hem utilitzar en la creació de la Toolbox les anomenarem a continuació i les descriurem breument.

- **char2cell :** Funció per convertir una cadena de caràcters en un cell string.
- **progressbar :** Utilitat per crear una barra de progrés.
- **mym:** Conjunt de funcions que permeten accedir a bases de dades a traves del Matlab.
- **M2HTML :** Conjunt de funcions que no estan incloses a la Toolbox però les hem utilitzat per generar la API automàticament.

3.1.3 Com s' ha adaptat el codi existent

No tot el codi que es va proporcionar inicialment s'ha pogut aprofitar directament. Cada un dels codis s'ha hagut d'estudiar, per entendre l'algorisme que aplica. Un cop entès l'algorisme aplicat s'han hagut de buscar els punts forts i els punts febles de cada una de les implementacions, en els casos que disposaven de duplicitat de codi o be plantejar-se noves implementacions que aportessin millores. Finalment s'ha hagut de valorar quina es la millor manera d'incloure totes aquestes funcionalitats al nostre projecte .

En alguns casos el codi existent s'ha pogut incloure a mode de llibreria externa com en el cas del VRE i no ha calgut fer hi cap tipus de modificació. Aquesta decisió s'ha pres perquè incloent el codi com si fossin unes llibreries independents de la Toolbox hem aconseguit que les futures modificacions en aquest codi funcionin sense fer cap modificació en el nostre codi. Cal esmentar que intentar incloure VRE a PCA era inviable, VRE és una tècnica massa complexa per plantejar-se incloure-la a PCA. Les funcions char2cell i progressbar també s'han inclòs directament al codi sense cap modificació.

D'altres parts del codi s'han hagut d'adaptar per tal de crear el nou codi. La part de CBR les funcions de retrieve i reuse s'han adaptat al funcionament en OO i s'han optimitzat conseqüentment per tal d'aprofitar tota la potencia que ens permet Matlab. També hi hem inclòs nous mètodes de votting i noves distancies.

Altres parts del codi, al ser algorismes simples o de dimensions més reduïdes, ha resultat més fàcil implementar de nou aquestes parts que adaptar-les al nou sistema de treball orientat a objectes. L'anàlisi de contribucions, la exportació de dades al weka , l'algorisme de cross validation, els mètodes de Drop4 i IB3 són les parts dels codis existents prèviament que han estat reescrits completament.

3.2 Llenguatge i Metodologia

Per tal de desenvolupar el projecte s'ha utilitzat el llenguatge de programació de Matlab tal com s'indica als objectius del PFC. A la Figura 14 hi tenim el logotip característic de Matlab.

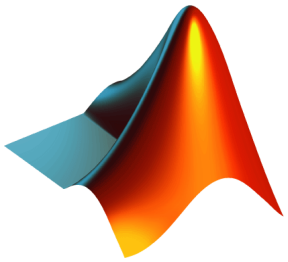


Figura 14

El fet de treballar en Matlab ens aporta avantatges com per exemple:

1. Poder reutilitzar els diferents codis de la gent del grup i integrar fàcilment les millores que s'estan fent actualment
2. Velocitat i facilitat d'utilització de càlculs Matricials
3. Accés a Toolbox addicionals (PLS Toolbox, Toolbox estadística), en cas de disposar de Toolbox addicionals les utilitzarem. (La Toolbox estadística és més eficient en el càlcul de components principals.)
4. Simplificació de la visualització de resultats. La creació de gràfics de tota mena és molt simple en Matlab.

El fet de treballar en Matlab ens limita en altres aspectes:

1. Matlab és un llenguatge majoritàriament imperatiu. En les ultimes versions es permet una major orientació a objectes i l'utilitzarem tant com ens sigui possible.
2. En Matlab, quan es treballa amb objectes es passa sempre per paràmetre l'objecte al què ens estem referint. Així doncs, la crida x.get() passarà a invocar el mètode get(x) de manera automàtica.

Pel que fa a la metodologia de programació que hem portat a terme, tot i que no

de manera estricte, és l'Extreme Programming. Aquesta metodologia és molt propera a l'usuari final i ens ha anat bé sobretot per anar incloent noves modificacions a la interfície d'usuari i aconseguir així un resultat satisfactori per als usuaris finals.

3.3 Requisits del sistema

La Toolbox i els diferents components que la formen han estat desenvolupats en Matlab 2010a. S'ha aprofitat que en les darreres versions del producte s'ha incorporat la possibilitat de programar amb Orientació a Objectes (OO).

Programar amb OO en Matlab ens aporta avantatges importants a l'hora d'encapsular el codi i determinar els permisos d'accés als atributs dels nostres objectes. Per altra banda, el principal inconvenient d'utilitzar la OO és que necessitarem per poder executar el codi una versió del Matlab 2008a o posterior.

Respecte als requisits mínims del sistema, no necessitem rés més que un equip capaç d'executar el Matlab. Tot i això, és molt recomanable disposar d'un equip superior als mínims establerts per tal d'assegurar una bona resposta i visualitzacions satisfactòries.

Els requisits mínims per poder executar el Matlab 2008a són:

- **CPU**: Pentium 4 o AMD Athlon 64
- **Espai al disc dur**: 510 MB lliures per instal·lar el Matlab
- **RAM**: 512 MB
- **Sistema operatiu**: Windows XP , Devian 4.0, Fedora Core 4, Red hat Enterprise Linux v.4 , Mac OS X 10.4
- **Tarja gràfica**: 16 bits compatible amb OpenGL

Característiques recomanades:

- **CPU**: 64bits amb 2 o més nuclis

- **RAM**: 4 o 8 GB de RAM
- **Tarja gràfica**: Compatible amb CUDA.

És molt recomanable tenir una resolució de pantalla mínima de 1024 x 768 per poder visualitzar sense distorsions les pantalles més grans.

3.4 Estructura general de la Toolbox

Com ja s'ha mencionat anteriorment a la planificació inicial, aquest projecte el podem dividir en 3 parts. Les llibreries, les interfícies i l'exemple d'aplicació.

A les llibreries és la part del codi on es fan els càlculs i a on hi ha implementats els algorismes més importants del nostre projecte. Les interfícies aporten a la Toolbox un seguit d'eines perquè els usuaris puguin invocar les funcions de la llibreria de manera simple i visual, el fet de tenir interfícies a la Toolbox fa que aquesta pugui funcionar a mode d'aplicació autònoma. Finalment l'exemple d'aplicació consta d'un cas d'aplicació de les llibreries en un escenari real aquí s'utilitzen totes les eines que proporciona la Toolbox per resoldre el problema que es planteja.

3.5 Disseny de les llibreries

És la primera part del PFC que va caldre afrontar. Les llibreries han de permetre com hem comentat als objectius aplicar PCA i CBR.

No és difícil adonar-se que els dos algorismes són bastant complexes pel que fa a la diversitat de configuracions que permeten, això fa que implementar-los com a una única funció sigui pràcticament inabastable.

Arribats a aquest punt se'ns van plantejar dues opcions a l'hora d'enfocar la creació de les llibreries, orientar el codi a funcions o bé utilitzar l'OO.

Tots els codis que disposàvem d'altres membres del grup utilitzen la programació orientada a funcions, segurament perquè la OO ha estat introduïda recentment al Matlab i fins fa relativament poc presentava molts problemes i encara ara presenta limitacions.

Treballar amb funcions obliga que l'usuari de les funcions conegui el format de les dades sobre les quals treballa la funció. Degut a la complexitat que tenen les nostres dades pot resultar difícil d'entendre el format de les estructures que es generarien, ens cal guardar moltes dades per evitar càlculs innecessaris.

Per tal d'evitar aquests problemes hem plantejat una solució mitjançant OO. Tot i que aquesta solució serà sempre menys eficient que la versió equivalent en format de funcions (problema inherent de la OO en Matlab), aquesta solució presenta grans avantatges pel que fa a la facilitat d'encapsulació del codi. Això ens permetrà abstracció de certs aspectes com el format intern de les dades per a l'usuari final.

El fet de treballar amb objectes també ens permet utilitzar funcions privades. Les funcions privades les utilitzarem per d'obtenir major eficiència i facilitar la detecció d'errors dividint els mètodes més complexes en dos nivells. El primer nivell, el mètode públic (el que invocaran els usuaris de la Toolbox) on es faran totes les comprovacions necessàries per tal d'assegurar que es pot executar el mètode privat (Comprovació dels paràmetres entrats per l'usuari, comprovació que tots els atributs que s'utilitzaran de l'objecte tinguin valor, informar d'errors,...). El segon nivell de funcions són els mètodes privats, que és on realment es faran els càlculs corresponents al mètode invocat per l'usuari. Aquesta divisió de funcionalitats també permetrà que internament puguem

invocar els mètodes privats i evitar-nos les comprovacions en els punts on no siguin necessàries.

La solució escollida és una solució mixta, utilitzarem l'OO en els 2 algorismes més complexes PCA i CBR i també per crear una estructura de dades DATA a la qual els usuaris nomes tinguin accés al que nosaltres volem. Per la resta de funcionalitats de la llibreria seguirem utilitzant la programació en funcions, ja sigui per evitar fer feina innecessària (VRE funciona perfectament, no cal reescriure el codi utilitzant l'OO), o be perquè són funcions molt simples i plantejar les com objectes les complicaria innecessàriament.

A continuació farem un petit resum de les funcionalitats assignades a cada un dels objectes que acabem d'esmentar. En els següents capítols detallaran més a fons aquests objectes i les relacions entre ells i la resta de funcions.

- **DATA** - És un objecte que s'encarregarà de guardar les dades del procés. Les seves funcionalitats també inclouen, guardar/carregar des de fitxers, conversions de VW i BW tant per l'entrada com sortida de dades.
- **PCA** - Aquest objecte permetrà aplicar un anàlisi de components principals sobre un objecte DATA. S'encarregarà d'estandarditzar les dades (segons el mètode que esculli l'usuari i el format que proporcionin les dades VW o BW), realitzar el càlcul de les components principals, escollir el nombre de components que s'utilitzaran (segons el mètode escollit), recuperar dades en qualsevol moment (Q, T^2, Límits...) i realitzar les gràfiques més comunes (T^2vsQ, Contribucions...).
- **CBR** - Aquest objecte rebrà un model PCA com a descriptor del problema i un objecte DATA a mode de base de casos. I permetrà a l'usuari aplicar el cicle característic d'aquesta tècnica (retrieve, reuse, revise, retain). També proporcionarà a l'usuari tècniques de neteja de la base de casos com DROP4, IB2, IB3,..

El model de classes de les llibreries és molt simple, classes pròpiament dites només en tenim 3, a la Figura 15 hi tenim el diagrama de classes.

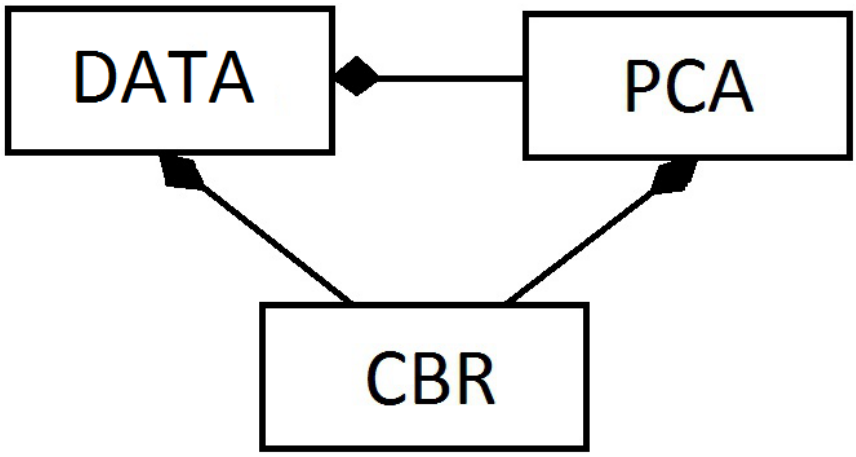


Figura 15

A més a més de les 3 classes com ja hem comentat anteriorment el codi de la llibreria consta de diversos paquets de funcions, si els entenem com a llibreries el diagrama de context de la llibreria es el que es mostra a la Figura 16.

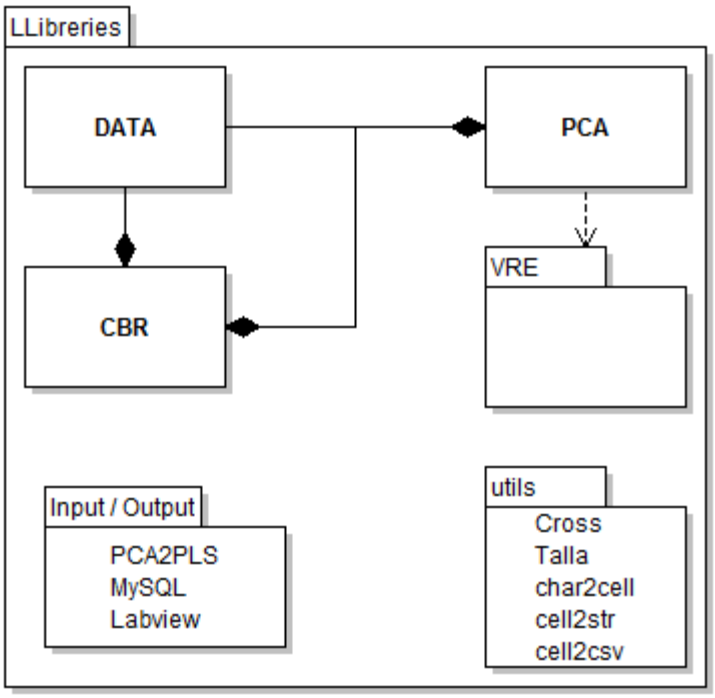


Figura 16

3.5.1 Classe DATA

La classe DATA és l'encarregada de muntar tota l'estructura de dades necessària per aplicar PCA i posteriorment, si es requereix, CBR. El codi d'aquest objecte és simple i ha de ser el màxim d'eficient possible ja que d'això en depèn l'eficiència global.

Les dades més importants emmagatzemades en aquest objecte són:

- El nom del procés emmagatzemat a l'objecte actual.
- Informació sobre les diferents fases.
- Informació sobre els lots de dades que tenim.
- Les dades del procés en format.
- La classificació i nom dels diferents lots.

Aquesta classe no nomes s'encarrega de guardar en memòria totes les dades del procés. Aquesta classe s'encarregarà de proporcionar un seguit de getters i setters a l'usuari per poder accedir o modificar les dades de l'objecte. El fet d'accedir a les dades a traves d'un mètode fa que puguem controlar les dades que tenim als atributs de l'objecte. També hi hem afegit altres funcionalitats directament relacionades amb les dades com fer els desdoblaments en batch-wise i variable-wise de les dades, permetre entrar dades desdoblades en batch-wise i variable-wise, persistència de l'objecte, sobre-escriptura de la visualització per defecte de Matlab de l'objecte al workspace.

3.5.1.1 Atributs

Els atributs de la classe DATA són privats. El fet de tenir els atributs privats ens permet controlar el tipus i la coherència de les dades que ens hi entra l'usuari i així evitar problemes a l'hora de fer els calculs. Encara que en Matlab els atributs d'una classe i les variables en general no tinguin un tipus ni mida associada, nosatlres obligarem amb els settersi metodes modificadors que els atributs tinguin les dimencions i tipus descrits a continuació.

string : nom

Aquest atribut és on guardem el nom del procés que ens entra l'usuari, no obligatori que un procés tingui nom, per tant aquest atribut serà Null sempre que no ens l'hagin especificat. Aquest atribut serà de tipus string, inicialitzant-se a null per defecte. La longitud del text emmagatzemat no la limitarem. Aquest valor es manté mentre l'usuari no el modifiqui amb el setter corresponent.

string[] : variables

En aquest atribut s'hi guarden els noms de les diferents variables que intervenen en el procés. Per facilitar el seu maneig aquest atribut serà un cellstring (un cell que conte text en totes les seves celles) d'una sola dimenció. Aquesta informació

serà important, sobretot, quan utilitzem les dades per fer PCA. Si no tenim aquesta informació, alguns dels dels grafics de control del PCA creat amb aquest objecte DATA podrien no mostrar tota la informació. Aquest atribut és doncs necessari i el nombre de variables (la longitud del vector), ha de ser el mateix que el nombre d'elements de la segona dimensió de l'atribut data.

string[] : fases

Aquest atribut guarda els noms de les diferents fases del procés. Per facilitar el seu maneig aquest atribut és un cellstring (un cell que conte text en totes les seves celles) d'una sola dimenció. Aquest vector ha de tenir el mateix nombre d'elements que durafases. Aquest atribut s'inicialitzarà a null en crear l'objecte i s'hi mantindrà mentre l'usuari no hi entri les dades amb el setter corresponent.

double[] : durafases

Aquest atribut és un vector de doubles i ens indicarà la duració de cadascuna de les fases. Lògicament la duració de la primera fase correspondrà a durafases[1], la de la segona a durafases[2] i així successivament. Aquest vector ha de tenir el mateix nombre d'elements que el vector de fases. Aquest atribut s'inicialitzarà a null en crear l'objecte i s'hi mantindrà mentre l'usuari no hi entri les dades amb el setter corresponent.

double : lots

Aquí es guardarà per evitar calcular-lo constantment el nombre de lots que té el procés. Aquest paràmetre s'actualitza de forma automàtica en el moment d'entrar les dades (batch-wise, variable-wise o matriu 3D), així com quan s'utilitzin mètodes que modifiquen el nombre d'aquests elements. Aquest atribut s'inicialitzarà a 0 en crear l'objecte i s'hi mantindrà mentre no tinguem dades a l'objecte. En cas d'eliminar tots els lots de l'objecte, aquest valor automàticament tornarà a ser 0.

string[] : reflots

Aquest atribut és on guardarem el nom o referència de cada un dels lots. Aquest paràmetre és opcional, si treballem amb la classe CBR ens serà molt útil per identificar els lots. Aquest metode d'identificació l'hem hagut d'incloure perque de la posicio no ens en podem fiar ja que varia. Aquesta llista sofrira les mateixes modificacions que les dades mantinguent sempre els lots identificats correctament, es a dir el nom sempre estarà a la mateixa posició que el lot a què fa referència.

double[][][] : data

L'atribut data és on guardarem les dades del procés (internament és una matriu 3D de doubles) i quan calgui en farem el desdoblament a BW o VW segons convingui.

double[] : class

És un vector de doubles on es guarda la classe de cada lot en format numèric. És molt important per assegurar un bon comportament de l'aplicació que aquest atribut tingui tants elements com lots hi hagi a les dades. Aquest atribut s'inicialitzarà a null en crear l'objecte i s'hi mantindrà mentre l'usuari no hi entri les classes dels lots amb el setter adient.

3.5.1.2 Mètodes

A continuació descriurem perquè serveixen els mètodes més rellevants de l'objecte DATA. Els getters i setters els obviarem perquè el seu propòsit és clar.

getRawData

Tal com indica el seu nom aquest mètode serveix per recuperar totes les dades de l'objecte DATA. Es poden recuperar les dades desdoblades en qualsevol dels 2 formats Bach wise o Variable wise. Per indicar el format desitjat utilitzarem el paràmetre d'entrada. Si li indiquem 0 o 'BW' les dades es retornaran en format

Bach wise pel contrari si indiquem 1 o 'VW' Les dades es retornaran en format Variable wise.

shuffle

Mètode que ens permetrà ordenar o desordenar els lots de la base de casos. Si no indiquem les posicions on volem els lots les posicions seran generades aleatòriament. Aquest mètode retornarà per cada lot la posició on estava anteriorment, amb aquest índex podrem desfer l'ordenació.

Save / load

Aquests mètodes s'encarregaran de la persistència de l'objecte. Permetran que l'usuari desi o carregui dades a l'objecte des de diversos formats. Propis de Matlab o definits expressament per la classe DATA.

Disp / openvar

Són sobrecarregues dels mètodes per defecte de Matlab per visualitzar i obrir objectes per editar-los. D'aquesta manera controlem com Matlab mostra i obra els nostres objectes.

3.5.2 Classe PCA

La classe PCA serà l'encarregada de mantenir en memòria les dades necessàries per fer l'anàlisi de components principals. Aquesta classe també permet la projecció de objectes DATA sobre el model PCA emmagatzemat a més d'altres mètodes útils com poden ser gràfics de control.

3.5.2.1 Atributs

Els atributs de la classe PCA són privats. El fet de tenir els atributs privats ens permet controlar el tipus i la coherència de les dades que hi entra l'usuari i així evitar problemes a l'hora de fer els calculs. Encara que en Matlab els atributs

d'una classe i les variables en general no tinguin un tipus ni mida associada, nosatlres obligarem amb els settersi metodes modificadors que els atributs tinguin les dimencions i tipus descrits a continuació.

DATA : dades

Objecte tipus DATA on hi ha el conjunt de dades inicials per crear el model PCA. És més, si no disposem de l'objecte DATA, no podem crear un objecte PCA.

bool : estàndard

Booleà que indica si les dades s'han estandaritzat (estàndard = true) o no (estàndard = false). Per defecte el valor s'inicialitza a false.

bool : model

Booleà que indica si la instància representa un model PCA (model = true), o si per contra, és una projecció a un model PCA (model = false). Les principals diferències entre un model i una projecció són les següents:

- Els models es creen amb el constructor de la classe (PCA::PCA()), mentre que les projeccions s'obtenen amb el mètode PCA::projecta.

- Els models PCA calculen la seva mitjana, desviació estàndard i límits estadístics, mentre que les projeccions les hereten del model PCA sobre el que s'han projectat.

double[][] : Z

Matriu bidimensional de doubles amb les dades estandaritzades. Per defecte s'inicialitza a null.

double[][] : P

Matriu bidimensional de doubles on guardem la matriu de components principals. Ha de ser igual al nombre de columnes de P. El seu valor per defecte és null.

double : Num

Nombre de components principals amb que s'ha creat el model. Ha de ser igual al nombre de columnes de P. El seu valor per defecte és null.

double[][] : T

Matriu de doubles on guardem la projecció de Z sobre el model PCA. Ha de tenir tantes files com Z i tantes columnes com P. El seu valor per defecte és null.

double[] : Lambda

Vector que conté els valors propis associats a les components principals retingudes. Ha de tenir tants elements com columnes té P. El seu valor per defecte és null.

double[] : Lambdai

Vector que conté els valors propis associats a totes les components principals. Té el nombre d'elements igual al rang de Z.

double[] : sta

Vector de doubles que conte la desviació de cadascuna de les variables mesurades del procés. Té tants elements com columnes té Z.

double[] : mea

Vector de doubles que conte la mitjana de cadascuna de les variables mesurades del procés. Té tants elements com columnes té Z.

double[][] : E

Matriu de doubles que conté la matriu residual de projectar Z sobre el model PCA.

double[] : T2

Vector de doubles que conte els valors de l'índex T^2 de cada observació en Z.

Te tants elements com files té Z.

double[] : LimT2

Double que indica el límit de T^2. qualsevol observació amb un valor de T^2 per sobre d'aquest límit es considera defectuosa per culpa de T^2.

double[] : LimQ

Double que indica el límit de Q. qualsevol observació amb un valor de Q per sobre d'aquest límit es considera defectuosa per culpa de Q.

double : metode

Double que identifica el mètode de selecció del nombre de components principals que s'utilitzarà. Els possibles valors són:

- **0 = Manual** - Demanem a l'usuari per consola que ens entri el numero de components que vol utilitzar.
- **1 = Kaiser (opció per defecte)** - Descarta totes les components que reconstrueixen menys que una variable inicial (lambda < 1).
- **2 = Colze** - Mostrem la gràfica dels valors propis i demanem a l'usuari que entri el punt on hi ha el colze.
- **3 = Colze + punt recomanat** - Com el mètode anterior, però recomana el un punt on s'ha detectat el colze.
- **4 = % de variància explicada** - Demanem a l'usuari el % de variància que vol que quedi explicada al model i escollim el nombre de components principals perquè es compleixi.
- **5 = VRE** - Aplica el mètode extern de *(Variance of the Reconstruction Error)* per determinar les variables i components principals a retenir.

double : metodee

Mètode d'estandardització que s'utilitzarà (per més informació vegeu 2.5.2). Els

valors vàlids són:

- **1=Auto scaling (opció per defecte)**
- **2=Grup scaling**
- **3=Continous scaling**
- **4=Block scaling**

bool : treuFM

Booleà que indica si volem treure la forma mitja de les dades (treuFM = true, valor per defecte), o no (treuFM = false). El fet de treure la forma mitja afecta si estem treballant en VW o en BW i s'aplica el Continuous scaling.

bool : conv

Booleà que fixa el mètode de desdoblar la matriu tridimensional de dades. conv = false desdobla en BW (opció per defecte), mentre que conv = 1 desdobla en VW.

VRE_Struct : struct

Estructura de dades on es guarda tota la informació relacionada amb el mètode del VRE.

double[] : CLimQstd

Vector de doubles que conté la desviació estàndard de les contribucions de Q. S'utilitza per al càlcul del límit d'aquestes contribucions.

double[] : CLimQmea

Vector de doubles per emmagatzemar la mitjana de les contribucions de Q. S'utilitza per al càlcul del límit d'aquestes contribucions.

double[] : CLimT2std

Vector de doubles que conté la desviació estàndard de les contribucions de T^2. s'utilitza per al càlcul del límit d'aquestes contribucions.

double[] : CLimT2mea

Vector de doubles que conté la mitjana de les contribucions de T^2. S'utilitza per al càlcul del límit d'aquestes contribucions.

3.5.2.2 Mètodes de modelat

A PCA s'han implementat mètodes de manera que es permet fer la selecció de components principals de diversos mètodes. Els mètodes de selecció de components que tenim implementats actualment són, Manual, Kaiser-Guttman, Percentatge de variància explicada, Cattell's Scree Test i VRE (Codi extern).

A més a més s'ha automatitzat el mètode Cattell's Scree Test que consisteix en trobar el punt d'inflexió de la gràfica que resulta al graficar les Lambdas (valors propis contra les components principals), un exemple d'aquesta gràfica el podem observar a la Figura 17 (s'ha fet un zoom en la zona d'interès).

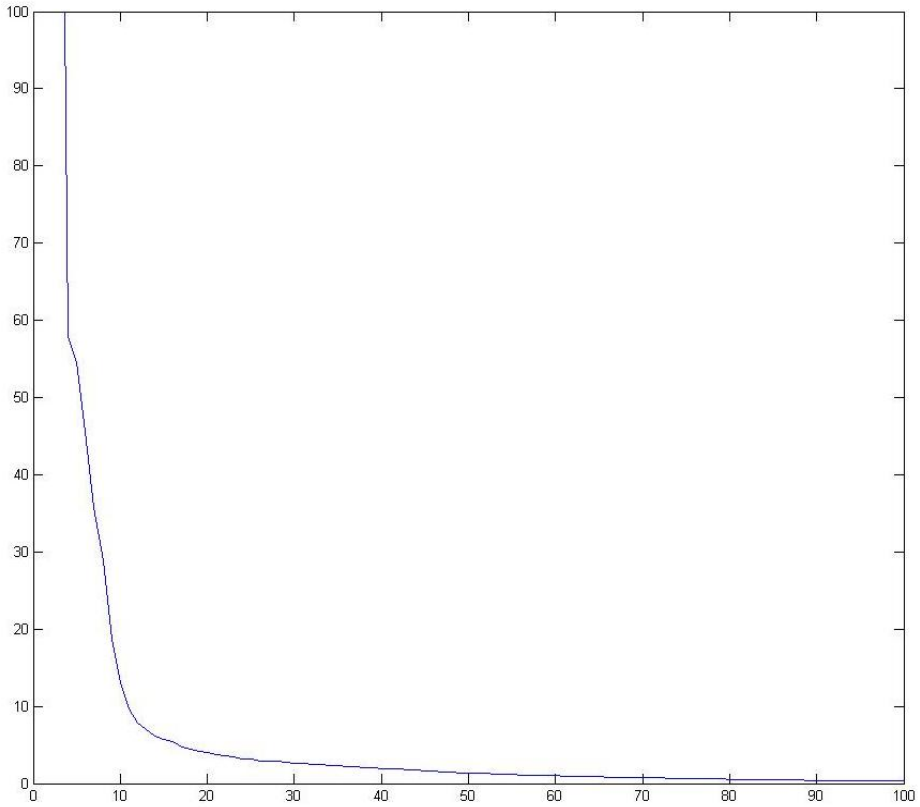


Figura 17

En la Figura 17 observem que el numero de components principals a escollir segons el criteri de Cattell's Scree Test seria 11, però depenent de l'usuari pot variar ja que es subjectiu.

Per automatitzar el mètode hem utilitzar la distància entre la recta que formen dos punts continus de la funció i l'origen de coordenades. El mínim d'aquestes distàncies serà on hi ha el colze.

3.5.2.3 Mètodes de visualització

Per tal de visualitzar les dades dels objectes PCA s' ha dotat a la classe d'un seguit de mètodes de visualització. Aquests mètodes van des de mètodes per representar les components principals, signatures,... fins a la sobre escriptura de la visualització per defecte de l'objecte en la línia de comandes de Matlab. A continuació farem una breu descripció dels mètodes més rellevants de visualització que s'han incorporat.

Disp / openvar

Són sobrecarregues dels mètodes per defecte de Matlab per visualitzar i obrir objectes per editar-los. D'aquesta manera controlem com Matlab mostra i obra els nostres objectes. Amb aquestes funcions hem fet que quan l'objecte hagi de ser mostrar per consola se'ns mostri informació útil sobre l'objecte com nombre de lots, variables, temps i les classes presents. També fan que en fer doble clic en un objecte PCA en el workspace s'obri una pantalla pròpia en comptes de l'editor de Matlab des d'on no es podria editar res perquè els atributs són tots privats.

plotMean/ plotStd

Són mètodes per visualitzar la mitja i la desviació estàndard que s'ha utilitzat per

estandarditzar les dades.

plotT2Q/ plotScreePlot/ plotLoadings/plotP

Són els gràfics de control més característics de PCA. Els hem preparat per que funcionin independentment del nombre de variables, classes o fases que tinguem a les dades.

plotQContribution/plotQMeanContribution/plotQSignatures

plotT2Contribution/plotT2MeanContribution/plotT2Signatures

Són els gràfics de control relacionats amb l'ús de les contribucions i les signatures. En el cas de les contribucions a més de tenir en compte de que fos independent del nombre de variables, fases,... s'ha hagut de tenir en compte quin mètode utilitzem per calcular les contribucions ja que canviarà el gràfic .

3.5.3 Classe CBR

Aquesta Classe utilitza les dues descrites en els capítols anteriors anteriors (PCA i DATA). L'objectiu de la classe CBR és, tal com indica el seu nom, aplicar tècniques de raonament basat en casos (*Case Based Reasoning*). Aquesta classe permetrà l'aprenentatge a partir d'exemples i la posterior recuperació i reutilització d'aquesta informació per resoldre casos similars. L'algorisme CBR implementat utilitzarà Objectes de tipus DATA per emmagatzemar els casos i objectes PCA per tal de trobar les variables mes rellevants del problema.

3.5.3.1 Atributs

Els atributs de la classe PCA són privats. El fet de tenir els atributs privats ens permet controlar el tipus i la coherència de les dades que hi entra l'usuari i així evitar problemes a l'hora de fer els calculs. Encara que en Matlab els atributs d'una classe i les variables en general no tinguin un tipus ni mida associada,

nosatlres obligarem amb els settersi metodes modificadors que els atributs tinguin les dimencions i tipus descrits a continuació.

PCA : Model

Objecte de tipus PCA que conté la informació del model on projectar els casos de la base de casos.

PCA : Case_Base

Objecte de tipus PCA, amb les projeccions dels casos de la base de casos sobre el model PCA. Aquest atribut és la base de casos en la què s'aplicaran els mètodes del CBR (retrieve, revise, reuse i retain).

double[] : vk

Aquí guardarem el paràmetre d'entrada "vk" descrit detalladament a l'apartat anterior d'aquest manual. El fet de guardar-lo ens permet que l'usuari pugui ometre'l en cas de repeticions.

double : dc

Aquí guardarem el paràmetre d'entrada "dc" descrit detalladament a l'apartat anterior d'aquest manual. El fet de guardar-lo ens permet que l'usuari pugui ometre'l en cas de repeticions.

double : method

Aquí guardarem el paràmetre d'entrada "method" descrit detalladament a l'apartat anterior d'aquest manual. El fet de guardar-lo ens permet que l'usuari pugui ometre'l en cas de repeticions.

bool : block

Booleà que indica si els paràmetres vk, dc i method estan bloquejats perquè s'ha fet una optimització de la base de casos (block = True) o no (block = False).

PCA : test

Objecte de tipus PCA que conté l'últim conjunt set de dades tractat o a tractar en breu.

struct : veins

Estructura que conté la informació dels veïns més propers recuperats. Aquesta estructura conté els següents camps:

- **indexs_k1 :** Posició relativa dels veïns més propers dins de la base de casos(ordenats de menor a major). La informació continguda en aquest nivell s'usa pel primer criteri de les distàncies combinades.
- **distancia_k1 :** Matriu de doubles amb la distància dels veïns retinguts per a cadascun dels nous casos entrats. Els valors estan ordenats de menor a major per cadascun dels casos nous (columnes de la matriu) respecte als veïns més propers recuperats de la base de casos(fila).
- **indexs_k2 :** Posició relativa dels veïns més propers dins de la base de casos(ordenats de menor a major). La informació continguda en aquest nivell s'usa pel segon criteri de les distàncies combinades i per les distàncies simples.
- **distancia_k2 :** Matriu de doubles amb la distància dels veïns retinguts per a cadascun dels nous casos entrats. Els valors estan ordenats de menor a major per cadascun dels casos nous (columnes de la matriu) respecte als veïns més propers recuperats de la base de casos(fila).

double [] : classepredit

Vector de doubles que conté la classe predita dels casos nous basant-nos en la informació dels veïns més propers.

3.5.3.2 Mètodes

A continuació descriurem perquè serveixen els mètodes més rellevants de l'objecte CBR. Els getters i setters els obviarem perquè el seu propòsit és evident.

Retain/Retrieve/Reuse/Revise

Són els mètodes característics de CBR. Sovint són anomenats les 4R.

IB2/IB3/DROP4

Són els mètodes de manteniment de la base de casos més habituals. Els hem implementat de la manera més eficient possible ja que són mètodes costosos computacionalment i que cal executar sovint.

IB_UdG

Mentre estudiàvem i incloíem els algorismes de neteja de la base de casos (IB2,IB3,DROP4) al nostre codi se'ns va acudir una manera d'aconseguir resultats similars però en ordre lineal respecte el nombre de casos de la base de casos. Cal mencionar que IB2, IB3 iDROP4 són d'ordre quadràtic respecte el nombre de casos existents a la base de casos.

El DROP4 té com a objectiu despoblar els centres de les classes i quedar, se amb les fronteres, d'aquesta manera es poden classificar correctament els mateixos casos que abans amb menys casos. Això és degut a que els casos es classifiquen mitjançant els seus veïns (Casos semblants, és a dir a poca distància).

El nostre mètode que hem anomenat IB UdG intentarà definir les fronteres de les classes i despoblar els centres. Per fer-ho, es busca el cas amb una classe diferent més proper a cada cas de la base de casos, que anomenarem enemic. L'enemic s'afegeix a la base de casos optimitzada, sempre i quant no estiguin ja inclosos a la base de casos final.

Per tal d'acurar els resultats es permet modificar el nombre d'enemics que s'afegeixen a la base de casos optimitzada per cada cas. El valor recomanat és el nombre mínim d'elements que es recuperen en fer un retreave. Si s'afegeixen un nombre d'enemics inferior a aquest valor, obtindrem una base de casos menor, però que al mateix temps, representa un major nombre de classificacions incorrectes. Per altra banda, si el nombre d'enemics és superior al nombre mínim de casos que recuperem, llavors tindrem una base de casos més gran, però que al mateix temps, tindrà un nombre de classificacions incorrectes més baix respecta la base de casos inicial.

També es permetrà descartar enemics per evitar el soroll a les fronteres intentant així separar-les. Ja que aquest mètode tendeix a acumular molts casos en les fronteres en problemes on hi ha soroll.

3.5.4 Altres Funcions

Tenim diverses funcions a la Toolbox, les quals formen part de la llibreria i que no hem inclòs en cap dels objectes. A continuació justificarem per cada una d'aquestes funcions per quin motiu s'ha decidit no treballar en OO i mantenir-les en format de funció, perquè serveix i quina relació te amb la resta de la llibreria.

VRE

La part del codi encarregada d'aplicar la tècnica del VRE és codi d'un altre membre del grup. En realitat són un conjunt de funcions que PCA utilitzarà si necessita aplicar VRE com a mètode de selecció de components principals. El fet de mantenir aquesta funcionalitat en un codi extern de PCA ens permet assegurar que el bon funcionament de PCA serà independent de la versió de llibreries de VRE utilitzada.

Talla

La funció Talla conceptualment podria ser un mètode de cada un dels objectes i

que s'invoquessin l'un a l'altre. Però queda molt més eficient i simple si s'engloba el codi en una única funció externa que pot rebre per paràmetre qualsevol dels 3 tipus d'objecte.

PCA2PLS

La funció PCA2PLS no està integrada a PCA perquè es volia que PCA fos una llibreria capaç de funcionar sobre Matlab sense cap Toolbox extra. Si hi ha Toolbox les fa servir com per exemple el cas de la comanda 'princomp' que és molt més eficient que la 'eig' però en el cas de la funció PCA2PLS es necessita tenir instal·lat la PLS Toolbox per poder-la utilitzar.

Cross4

Aquesta funció que anomenem Corss4 serà l'encarregada de realitzar un cross-validation sobre objectes de tipus DATA. Aquesta funció utilitzarà DATA, PCA i CBR per dur a terme el cross-validation segons la configuració que esculli l'usuari.

Mym

Són un conjunt de funcions que permetran a Matlab comunicar-se amb bases de dades MySQL. Aquestes funcions són extretes de Matlab file Exchange, també poden trobar-se a la pagina web oficial del projecte:

http://mym.sourceforge.net/

Aquest paquet de funcions inclou codi C++ que cal compilar amb Matlab per obtenir les llibreries que s'encarregaran de la comunicació amb MySQL.

Dpr

Funció simple que calcula la distancia entre un punt i una recta indicada per 2 punts. Aquests càlculs es necessiten a PCA per automatitzar el mètode de selecció de components screeplot. És més lògic fer els càlculs en una funció

externa perquè així tothom la podrà utilitzar.

Degradacolors

Una funció que utilitza PCA en alguns dels seus plots, aquesta funció construeix un degradat que comença amb el color pur i acaba a pràcticament blanc. Es poden indicar el nombre de colors a utilitzar (es seleccionaran tan distants com es pugui) i el nombre d'instants que te el degradat. Aquesta funció s'ha tret de PCA per permetre que altres usuaris la puguin reutilitzar fàcilment.

Cell2csv

Funció que converteix un cell, en un fitxer CSV.
El codi font que vam utilitzar com a vase es va extreure de:
http://www.mathworks.es/matlabcentral/fileexchange/4400-cell-array-to-csv-file-cell2csv-m

Sobre el qual vam arreglar alguns errors (solucionats a la versió actual).

Cell2str

Funció utilitzada per optimitzar el procés de creació dels datatips a la funció que crea les matrius de confusió. Tenim les dades que volem mostrar en cells i el datatip ha de ser un string aquesta conversió pot ser extremadament lenta si es fa amb un bucle,per això s'ha creat aquesta funció utilitzant la funció cellfun. Aquest procés pot ser utilitzat genèricament per convertir Cellstrings a strings i pertant el mantindrem com a funció autònoma.

char2cell

Funció extreta del File Exchange de Matlab. Més concretament el codi font original el podem trobar a:

http://www.mathworks.es/matlabcentral/fileexchange/24915-char2cell

Aquesta funció serveix per convertir un conjunt de caràcters en un cell indicant el caràcter que actua de separador i el que actua de salt de línia.

Plot_Bars

Funció que mostra un gràfic de barres amb el percentatge de calcificació de cada una de les particions fetes en el cross-validation.

Plot_Confusion_Matrix

Funció que crea una interfície visual, mitjançant un plot, per mostrar les matrius de confusió. Els càlculs de la matriu de confusió es poden passar per paràmetre, si no es fa es faran els càlculs amb les dades proporcionades en els paràmetres.

Progressbar

Funció extreta del File Exchange de Matlab. Més concretament el codi font original el podem trobar a:

http://www.mathworks.es/matlabcentral/fileexchange/6922-progressbar

Aquesta funció la utilitzem per mostrar el progres dels càlculs del procés de Cross-validation a l'usuari.

3.6 Disseny de la interfície

Un cop creades les llibreries, es va poder abordar la creació de un seguit de interfícies genèriques per tal de facilitar la utilització de les llibreries. Aquestes interfícies havien de ser genèriques (per qualsevol tipus de dades vàlid per la llibreria), fàcils d'utilitzar (en Matlab l'usuari sempre podrà invocar les funcions manualment, sense utilitzar les interfícies, si això resulta més fàcil que fer servir les interfícies la seva utilitat quedarà compromesa), permetre al màxim de llibertat a l'usuari.

3.6.1 Eines utilitzades

En Matlab les interfícies es creen amb la utilitat GUIDE. Les interfícies per defecte consten dos parts un '.fig' (una figura estàndard de Matlab) que és on es defineixen tots els botons i aspectes visuals de la interfície i un '.m' (un fitxer de codi estàndard de Matlab) amb totes les callbacks associades a la figura. En les interfícies no es possible aplicar la orientació a objectes i englobar en classes aquestes funcions. El motiu és que els Callbacks definits a la figura són invocats de manera automàtica en format de funcions (@(paràmetres)nom_de_la_figura), si s'engloben les funcions en objectes les crides automàtiques no invoquen el mètode corresponent, les callbacks simples com clicar botons es pot canviar fàcilment però hi ha funcions privades de Matlab que no es poden canviar, al menys a la versió 2010a que es la que hem utilitzat per desenvolupar les interfícies. A la versió 2012a han realitzat diversos canvis en els handles de les funcions sobretot en les interfícies gràfiques i possiblement ara en les versions més noves ja es pugui fer.

3.6.2 Patrons utilitzats en el disseny

Cal esmentar que per defecte les interfícies en Matlab són instàncies úniques, tot i que aquesta característica es pot deshabitar no ho hem fet. No permetre més d'una instància de cada interfície ajuda a limitar la llibertat de l'usuari. Si no fos perquè en realitat són funcions i estructures de dades, podrem entendre cada interfície com si haguéssim aplicat un patró singleton perquè el comportament final serà el mateix. I així seguim amb la OO que havíem iniciat en les llibreries.

3.6.3 Mobilitat entre Pantalles

Per tal de permetre una major mobilitat entre les interfícies es crearà a les pantalles més importants un menú que donarà accés a la resta de pantalles. Aquest element és el que fa que la nostra aplicació no tingui un flux definit i permet que l'usuari tingui certa llibertat en navegar per les pantalles. Tot i això per coherència, no es permetrà l'accés des de les interfícies a les pantalles que treballen amb objectes que no tinguem en el workspace, tot i això ha calgut preparar totes les interfícies per si són invocades des de comandes i no existeixen objectes del tipus adequat.

El diagrama d'activitats del menú és el que mostrem a la Figura 18. Com podem observar les interfícies no tenen un flux concret. Apareixeran segons el que vagi fent l'usuari. El fet de mantenir tenir les interfícies com a singletons ens obliga a comprovar si aquesta ja existeix abans de obrir una finestra, si és el cas no es crearà de nou simplement se li atorgarà el focus.

Tampoc podem oblidar que l'usuari d'aquestes interfícies estarà treballant en Matlab. Això bol dir que podrà fàcilment invocar comandes des de la consola, modificar els objectes o eliminar-los sense haver de utilitzar les interfícies ni funcions que nosaltres li proporcionem. Per tant haurem de comprovar constantment que la informació visualitzada en les interfícies continua essent valida. Aquesta peculiaritat de Matlab podem reflectir-la en UML considerant que cada usuari en realitat són 2 usuaris diferenciats i amb casos d'us diferents. En les versions compilades de la Toolbox no es té accés a la consola de Matlab i per tant no hi existirà el Comand user.

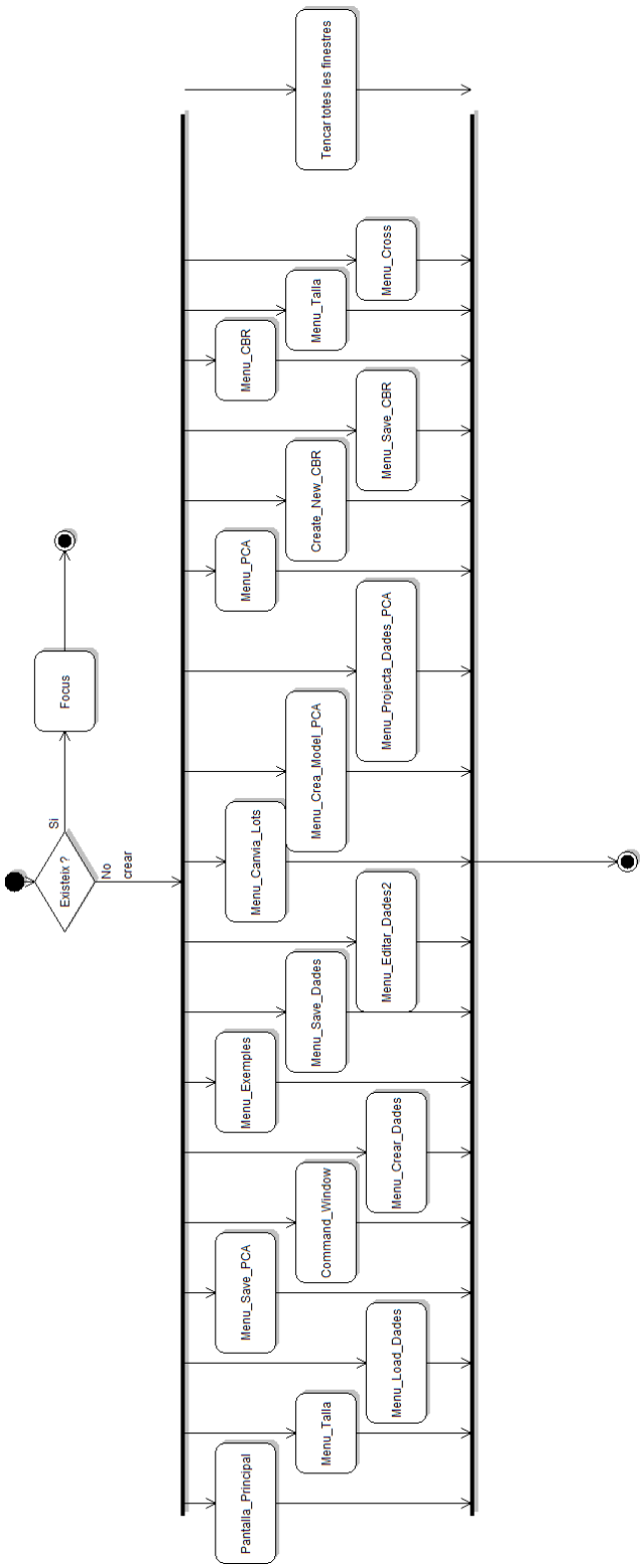

Tancar totes les finestres
Menu_Cross
Menu_Talla
Menu_CBR
Menu_Save_CBR
Create_New_CBR
Menu_PCA
Menu_Projecta_Dades_PCA
Menu_Crea_Model_PCA
Menu_Canvia_Lots
Menu_Editar_Dades2
Menu_Save_Dades
Menu_Exemples
Menu_Crear_Dades
Command_Window
Menu_Save_PCA
Menu_Load_Dades
Menu_Talla
Pantalla_Principal
Focus
Existeix ?
SI
No
crear

Figura 18


Executar comandes Matlab
Acces MySQL
Acces Labview
Command user
Usuari
«include»
«include»
Interface user
Crear objecte
Eliminar objecte
Segmentar objecte
Executar comandes Matlab simples
Guardar Objecte
Modificar objecte
Visualitzar objecte
Cross-Validation
Ajuda
Crear DATA
Des de Workspace
Des de Fitxer
«extend»
«extend»
«extend»
Crear PCA
New Model
New Projection
«extend»
«extend»
«extend»
«extend»
Crear CBR
«include»
«include»
«include»
«include»
Actualitzar interficies
DATA
PCA
CBR
«extend»
«extend»
«extend»
«extend»
«extend»
«extend»
«extend»
«extend»
Retrieve
Reuse
Revise
Retain
Optimitzar

Figura 19

3.6.4 Pantalles més importants

En aquest capítol s'explicarà el funcionament de les pantalles més interessants per il·lustrar la feina feta. Si es vol una descripció més detallada de totes les interfícies es podrà trobar al manual d'usuari a l'annex d'aquesta memòria.

3.6.4.1 Pantalla_Principal

Aquesta és la Pantalla inicial de l'aplicació. Des d'aquí es pot accedir a qualsevol de les altres pantalles directe o indirectament. Hi podem accedir executant l'script 'Pantalla Principal.m' des del Matlab, o bé executant l'executable de l'aplicació en cas d'utilitzar una versió compilada de la Toolbox.

Aquesta finestra té 2 funcionalitats bàsiques.

- És el punt de partida de l'aplicació.
- Permetre a l'usuari gestionar els objectes en memòria.

La Figura 20 mostra l'aspecte de la finestra. Amb un primer cop d'ull podem observar que a la part inferior hi tenim els logotips de l'UdG i del Grup eXiT , a la part superior el menú i al centre de la finestra on hi ha el gestor d'objectes.

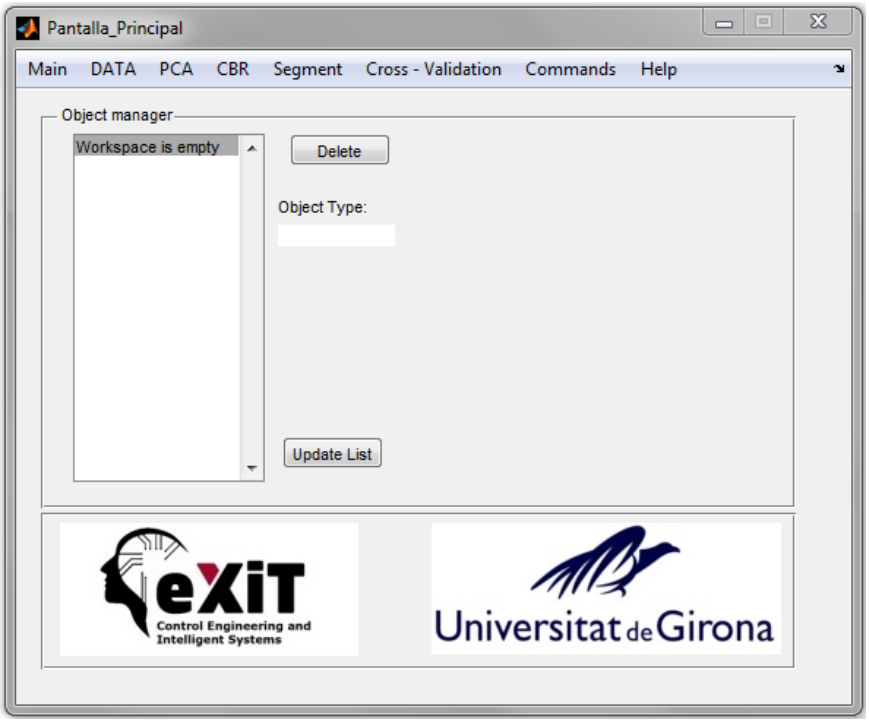


Figura 20

El gestor d'objectes:

Aquest element té l'aspecte que es mostra en la Figura 21.

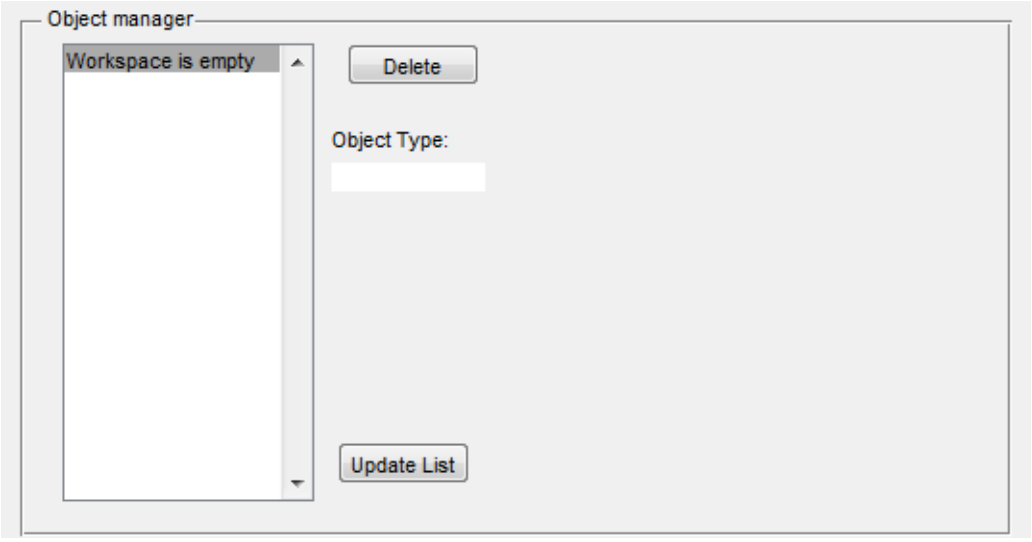


Figura 21

Aquí en el gestor d'objectes, a l'esquerra, podem observar una llista on es

mostren els objectes disponibles en el workspace. En el cas que el workspace estigui buit es mostrarà el missatge 'Workspace is empty' com a únic element de la llista. Si l'objecte que desitgem no és visible en aquesta llista segurament tampoc serà accessible per la resta de finestres. Si l'objecte ha estat creat mitjançant comandes possiblement es necessita actualitzar la llista prement el botó **Update List** perquè ens aparegui.

El botó **Update List** realitza una cerca d'objectes al workspace i actualitza la llista amb els elements trobats.

El botó **Delete** elimina del workspace l'element que hi hagi seleccionat a la llista.

En seleccionar un element de la llista ens apareixerà el tipus de l'objecte seleccionat en el display **Object Type**.

En fer doble clic sobre un element de la llista s'obrirà la pantalla corresponent per editar els paràmetres de l'objecte.

Els enllaços:

A la part inferior de la pantalla trobarem els logotips de la Universitat de Girona i del Grup eXiT. L'aparença d'aquest element és el que es mostra a la Figura 22. En clicar aquests logos s'obrirà la pagina web de la Universitat o del grup eXiT segons quin dels logotips s'hagi clicat.

Figura 22

3.6.4.2 Pantalla per treballar amb PCA

Aquesta pantalla és la que serveix per editar i visualitzar els objectes PCA. L'aspecte d'aquesta pantalla és el que es mostra en la Figura 23. En aquesta pantalla si pot accedir des del menú principal o executant l'script 'Menu_PCA.m'.

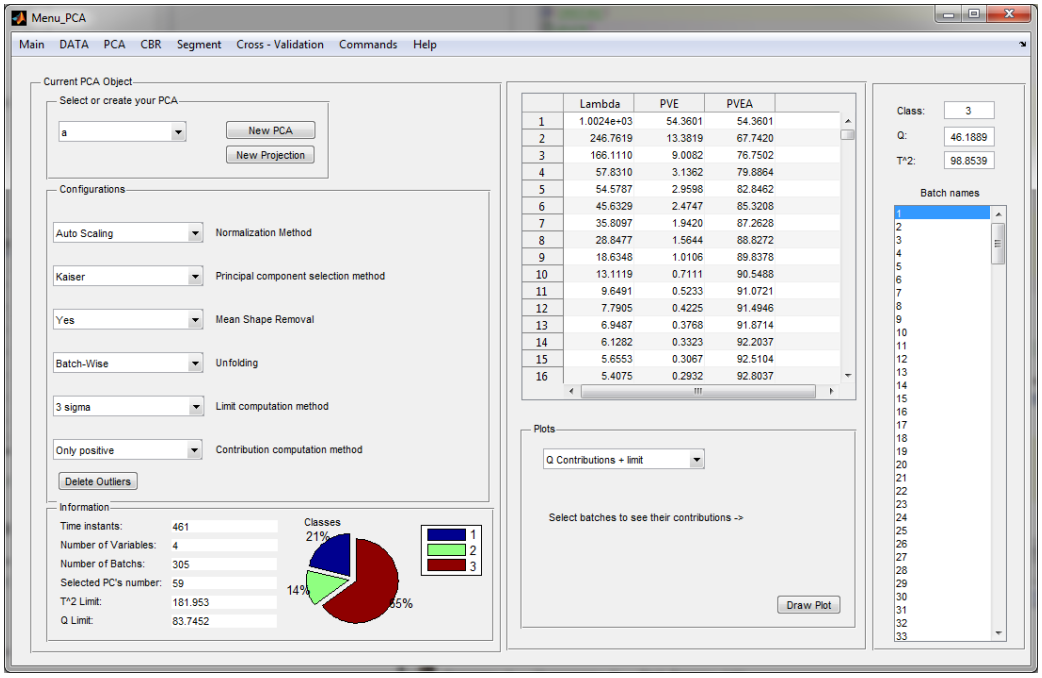


Figura 23

Aquesta pantalla està dividia en 6 apartats: Selecció de l'objecte amb què treballarem, Configuració, informació sobre l'objecte PCA, Selecció de components principals, plots i Informació sobre els lots.

<u>Selecció de l'objecte amb què treballarem:</u>

És el primer panell que cal utilitzar en aquesta pantalla, si l'objecte seleccionat per defecte no és el que volem. Per escollir l'objecte PCA amb el que volem treballar, utilitzant el panell **Select or create your PCA** (Figura 24), haurem de seleccionar l'element desitjat de la llista. En fer-ho les dades de la resta de panell

s'actualitzaran i mostraran les dades de l'objecte seleccionat.

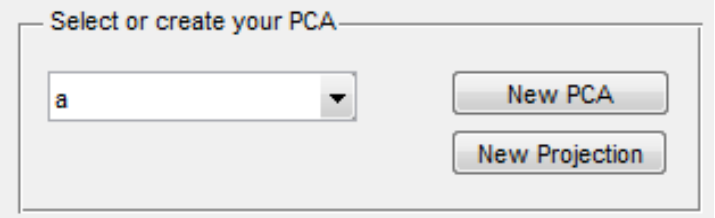


Figura 24

En aquest panell també es poden crear nous objectes, per crear un nou model PCA utilitzarem el botó **New PCA** per accedir a la pantalla de creació, per crear una nova projecció sobre un model PCA utilitzarem el botó **New Projection** per accedir a la pantalla on podrem crear-la.

Configuració:

En aquest panell es mostrarà en tot moment la configuració de l'objecte PCA, a més a més (sempre que l'objecte sigui un model) es podrà modificar la configuració. En modificar la configuració s'actualitzaran les dades de la resta de panells.

Les configuracions disponibles són les mateixes que s'han comentat en la pantalla configurar PCA amb un botó afegit per treure outliers.

El botó Delete Outliers serveix per treure els outliers del model actual. En prémer el boto apareixerà el diàleg que es mostra en la Figura 25 . L'opció **All** treurà tots els outliers del Model, l'opció Extream només treurà els outliers extrems.

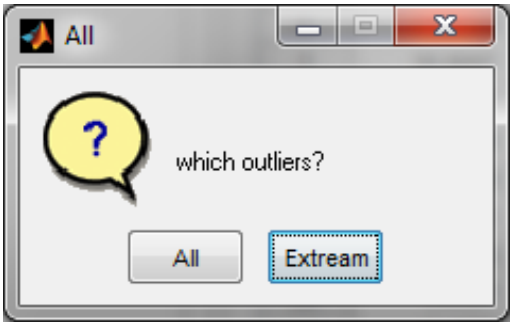


Figura 25

Informació:

En aquest panell, l'aspecte del qual es mostra a Figura 26, és on es mostraran les dades més importants sobre el Model. Aquestes dades s'actualitzaran tan bon punt es facin canvis en l'objecte mitjançant la pantalla actual.

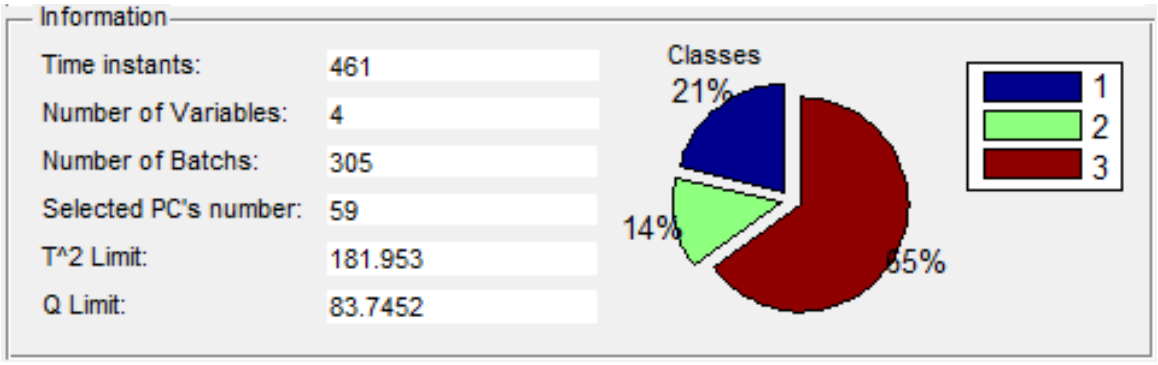
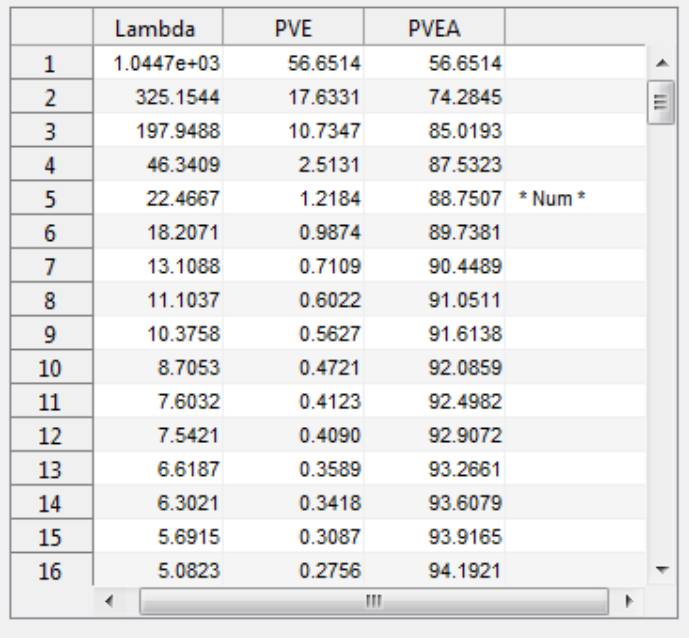


Figura 26

Selecció de components principals:

Aquest panell té una doble funcionalitat, informar i permetre escollir de manera fàcil el nombre de components principals a utilitzar en el model. Com es pot observar a la Figura 27 tenim 4 columnes, **Lambda** , **PVE**, **PVEA** i una última sense nom. La columna **Lambda** mostra el valor de Lambda de cada una de les components principals, la columna **PVE** mostra el percentatge de variància explicada per cada component, la columna PVEA mostrarà el Percentatge de variància acumulat, a la última columna es marcarà amb ***Num*** el nombre de components principals seleccionat.



	Lambda	PVE	PVEA	
1	1.0447e+03	56.6514	56.6514	
2	325.1544	17.6331	74.2845	
3	197.9488	10.7347	85.0193	
4	46.3409	2.5131	87.5323	
5	22.4667	1.2184	88.7507	* Num *
6	18.2071	0.9874	89.7381	
7	13.1088	0.7109	90.4489	
8	11.1037	0.6022	91.0511	
9	10.3758	0.5627	91.6138	
10	8.7053	0.4721	92.0859	
11	7.6032	0.4123	92.4982	
12	7.5421	0.4090	92.9072	
13	6.6187	0.3589	93.2661	
14	6.3021	0.3418	93.6079	
15	5.6915	0.3087	93.9165	
16	5.0823	0.2756	94.1921	

Figura 27

Per modificar el nombre de components principals es pot fer escollint un nou mètode de selecció del panell de configuració o be fent doble clic amb el botó secundari del ratolí sobre el nombre de components desitjat. Escollir el nombre de components manualment amb el ratolí provocarà que el mètode de selecció es canviï automàticament a **Manual** sempre es podrà tornar a escollir un nou mètode mitjançant el panell configuracions.

Plots:

En aquest panell es permetrà a l'usuari escollir d'entre diversos plots per visualitzar la informació de l'objecte actual. El funcionament d'aquest panell és molt simple, només cal escollir el plot desitjat de la llista, entrar els paràmetres que se'ns demanen i prémer el botó **Draw Plot**. A continuació detallarem per a cada un dels possibles plots quins paràmetres es demanaran a l'usuari i quin resultat s'espera obtenir.

Q contributions + Limit i T^2 contributions + Limit

En aquest 2 plots es mostraran les contribucions de Q o T^2 dels elements que s'hagin seleccionat de la llista de lots (**Batch Names**) situada en el panell de més a la dreta de la pantalla.

T^2 vs Q

No requereix d'inputs.

Scores

En seleccionar aquest plot ens apareixerà un requadre de text sota el selector de plots. Aquí se'ns demanarà que introduïm els números de les components principals que volem representar. Tenim diverses opcions segons el tipus de representació que es volgués obtenir:

- *numero* - Es representarà la component principal indicada respecte el temps.
- *numero, numero* - Es representarà la primera component indicada respecte la segona.
- *numero, numero, numero* - Es representars la gràfica tridimensional de les 3 components principals indicades.

% explication variable

No requereix d'inputs.

Q Signatures i T^2 Signatures

No requereix d'inputs.

Screeplot

No requereix d'inputs.

Mean

No requereix d'inputs.

Standard desviation

No requereix d'inputs.

P

En seleccionar aquest plot ens apareixerà un requadre de text sota el selector de plots. Aquí se'ns demanarà que introduïm els números de les P que volem graficar. Per introduir els valors podem enumerar-los en una llista [], entrar un valor o bé una expressió vàlida de Matlab.

Loadings

En seleccionar aquest plot ens apareixerà un requadre de text sota el selector de plots. Aquí se'ns demanarà que introduïm els números dels loadings que volem representar. Tenim diverses opcions segons el tipus de representació que volguem obtenir:

- *numero* - Es representarà el loading respecte el temps.
- *numero, numero* - Es representarà el loading indicant respecte la segona.
- *numero, numero, numero* - Es representarà la gràfica tridimensional dels 3 loadings indicats.

Variables

En aquest plot es mostraran els valors de les variables al llarg del temps dels elements que s'hagin seleccionat de la llista de lots (**Batch Names**) situada en el panell de més a la dreta de la pantalla.

Outliers Q or T^2

No requereix d'inputs.

Informació sobre els lots

En aquest panell es mostra la llista de noms dels lots de l'objecte, en aquesta llista podrem seleccionar elements i se'ns mostrarà a la part superior la Classe,

valor de Q i valor de T^2. També podrem seleccionar elements i clicant el botó secundari del ratolí eliminar lots, canviar-los el nom o la classe escollint l'opció adient del menú que ens apareixerà (Figura 28). Com ja s'ha comentat en l'apartat sobre el panell plots hi ha diversos plots que utilitzaran la selecció de lots en aquesta llista per saber quins lots mostrar.

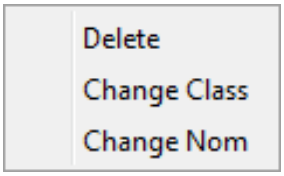


Figura 28

3.6.4.3 Pantalla per treballar amb CBR

Aquesta pantalla permet a l'usuari treballar amb les instàncies de CBR existents al workspace. L'aspecte d'aquesta pantalla és el que es mostra en la Figura 29. En aquesta pantalla s'hi pot accedir des del menú principal, fent doble clic en un objecte CBR o bé executant l'script 'Menu_CBR.m' sempre que hi hagi algun objecte CBR en el workspace.

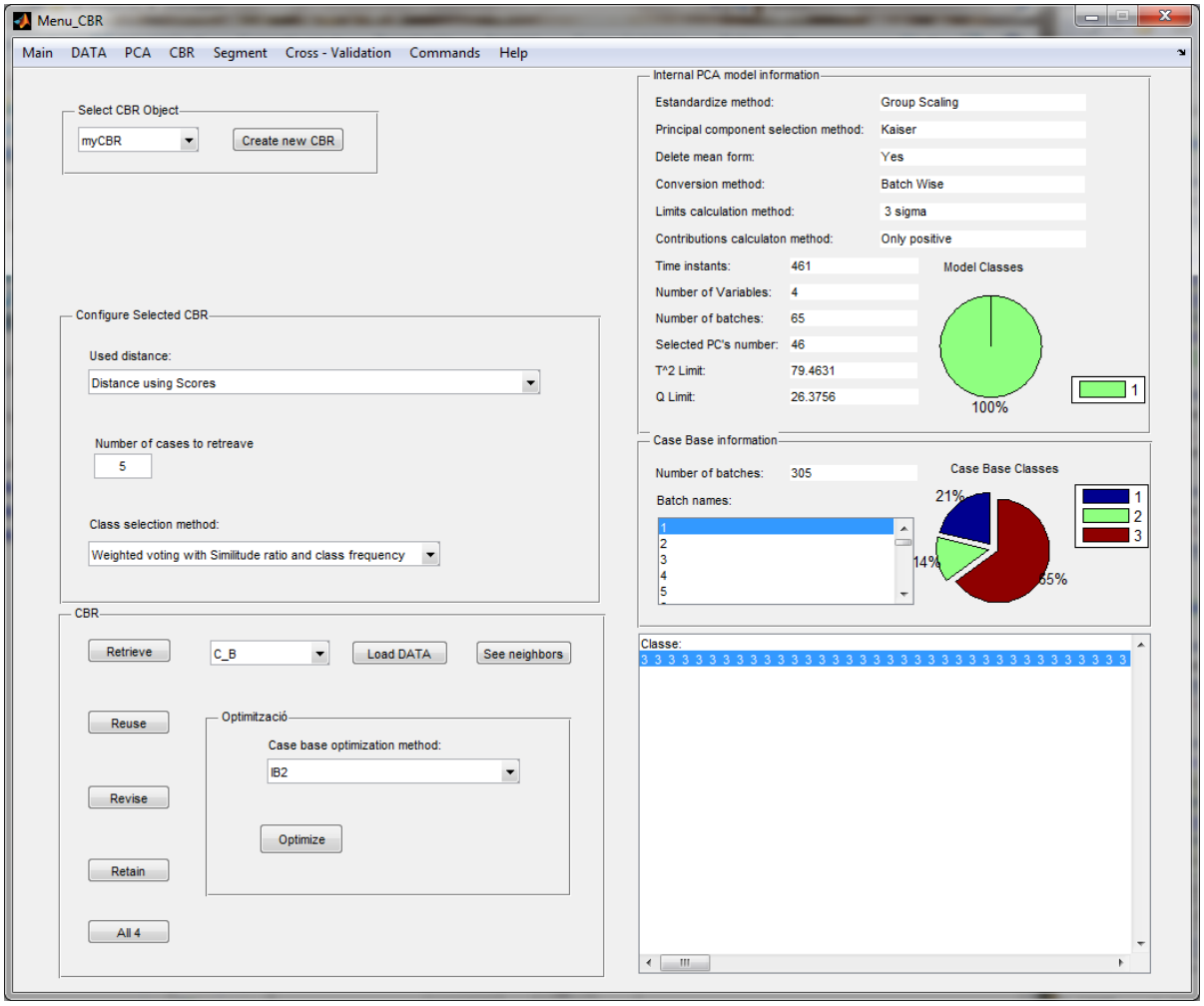


Figura 29

A continuació explicarem perquè serveix cada un dels panells que tenim en aquesta pantalla.

Select CBR Object:

El funcionament d'aquest panell (Figura 30) és molt simple. Serveix per seleccionar l'objecte CBR amb què volem treballar. Els objectes disponibles al workspace apareixeran a la llista. El botó **Create new CBR** el podrem utilitzar per accedir a la pantalla de creació d'objectes CBR.

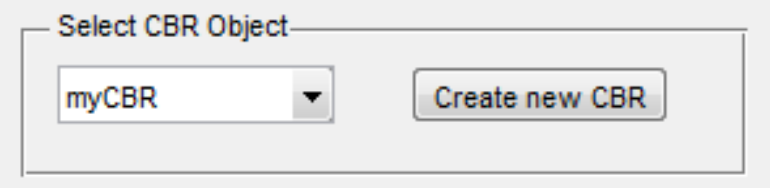


Figura 30

Configure Selected CBR:

Un cop seleccionat l'objecte amb el qual volem treballar en aquest panell se'ns permetrà configurar els paràmetres del retrieve i el reuse. Depenent de quina distància s'hagi escollit en el primer menú desplegable apareixeran més o menys camps per emplenar.

Les distàncies simples només necessiten com a imput el nombre de casos que torna el retrieve (Figura 31), pel que fa a les distàncies de 2 nivells necessiten els casos que s'ha de quedar en cada un dels nivells (Figura 32).

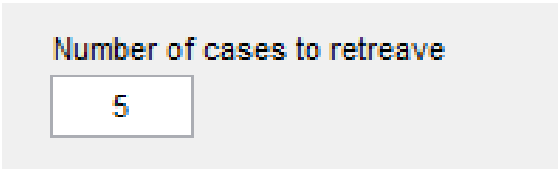


Figura 31

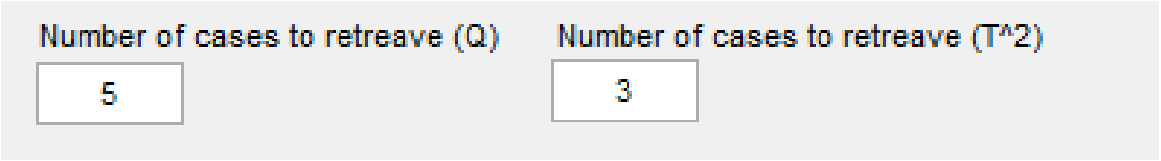


Figura 32

Hi ha algunes distàncies (**Distance using Q Signatures with binary phase agroupations i Distance using T^2 Signatures with binary phase agroupations**) que necessiten paràmetres extres (Figura 33). En aquests casos se'ns demanarà el nombre de casos que ha de retornar el retrieve i un paràmetre específic de la distància.

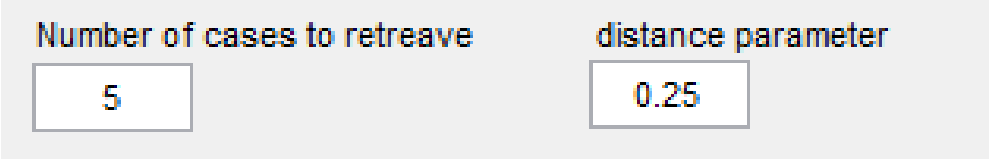


Figura 33

CBR:

Aquest panell és on l'usuari podrà invocar els diferents mètodes del CBR. Retrieve, reuse, revise, retain i optimize. Caldrà que s'esculli un conjunt de casos de test (un objecte DATA per projectar sobre el model contingut en l'objecte CBR). Els possibles casos de test apareixeran a la llista que hi ha al costat del botó retrieve. Si no apareix cap element o bé no hi ha el que es desitja, podrem accedir a la pantalla de creació d'objectes DATA amb el botó **Load DATA**.

Per realitzar les funcionalitats de Retrieve, reuse, revise i retain només caldrà prémer els botons amb aquests mateixos noms. Si es vol fer el cicle complert només cal clicar el botó **All 4**. En clicar cada un dels botons es mostraran els resultats (a la part inferior esquerra de la pantalla) i s'actualitzaran les dades de la part dreta de la pantalla.

Internal PCA model information:

En aquest panell (Figura 34) es mostra la informació útil sobre el model que té l'objecte CBR. Aquí podem trobar el nombre de casos que té, la configuració i els límits de Q i T^2.

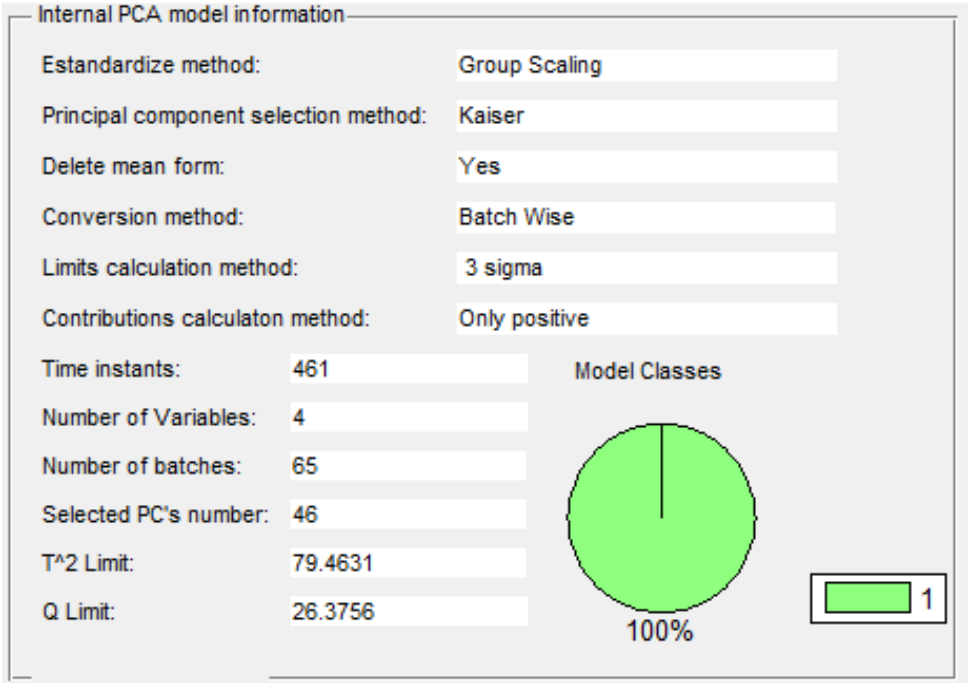


Figura 34

Case Base information:

En aquest panell (Figura 35) s'informa del nombre de casos, noms dels casos i percentatge de casos de cada classe de la base de casos de l'objecte CBR.

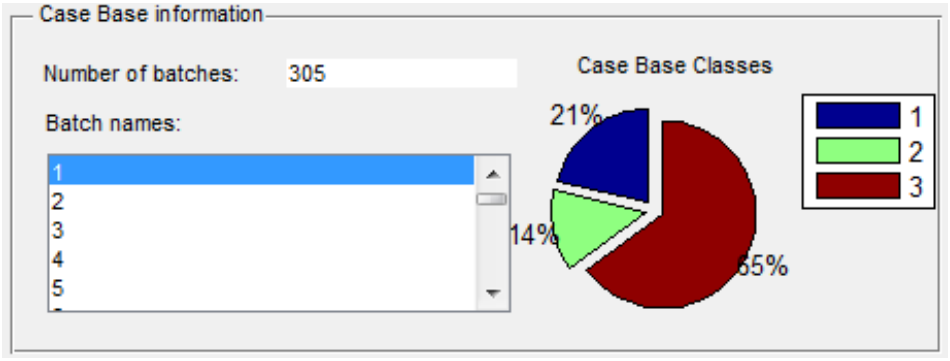


Figura 35

Resultats:

Aquest espai pot presentar diverses configuracions depenent de l'últim botó que s'hagi clicat.

- **Retrieve** - Mostrarà en un requadre de text la classe dels veïns i el nom de cada lot de test.
- **Reuse** - Mostrarà en un requadre de text la classe assignada a cada un dels casos de test
- **Revise** - Es mostrarà la matriu de confusió, on clicant al costat de cada nombre se'ns informarà dels noms dels lots que hi estan comptabilitats.
- **Retain** - Es seguirà mostrant la matriu de confusió, si hi ha canvis en la base de casos es mostraran actualitzant el panell **Case Base information**.
- **All 4** - Es mostrarà la matriu de confusió i si hi ha canvis en la base de casos es mostraran actualitzant el panell **Case Base information**.
- **Optimize** - Bloquejarà la modificació en el panell de configuracions, es mostraran en un requadre de text la classe dels veïns i el nom de cada lot de test, si fa falta s'actualitzarà el panell **Case Base information**.

Si el mètode d'optimització escollit és l'IB_UdG apareixeran 2 requadres per als paràmetres necessaris d'aquest mètode (Numero d'enemics a agafar i enemics que s'ignoraran).

3.6.4.4 Pantalla per Veure Veïns

Aquesta pantalla mostra la informació dels veïns més propers que s'han recuperat de la base de casos per a un seguit d'observacions. L'aspecte d'aquesta pantalla es mostra en la Figura 36. A aquesta finestra només si pot accedir des de les pantalles Menu_CBR (amb el boto **See neighbors**) i Menu_Cross (Mostrarà els resultats de les particions).

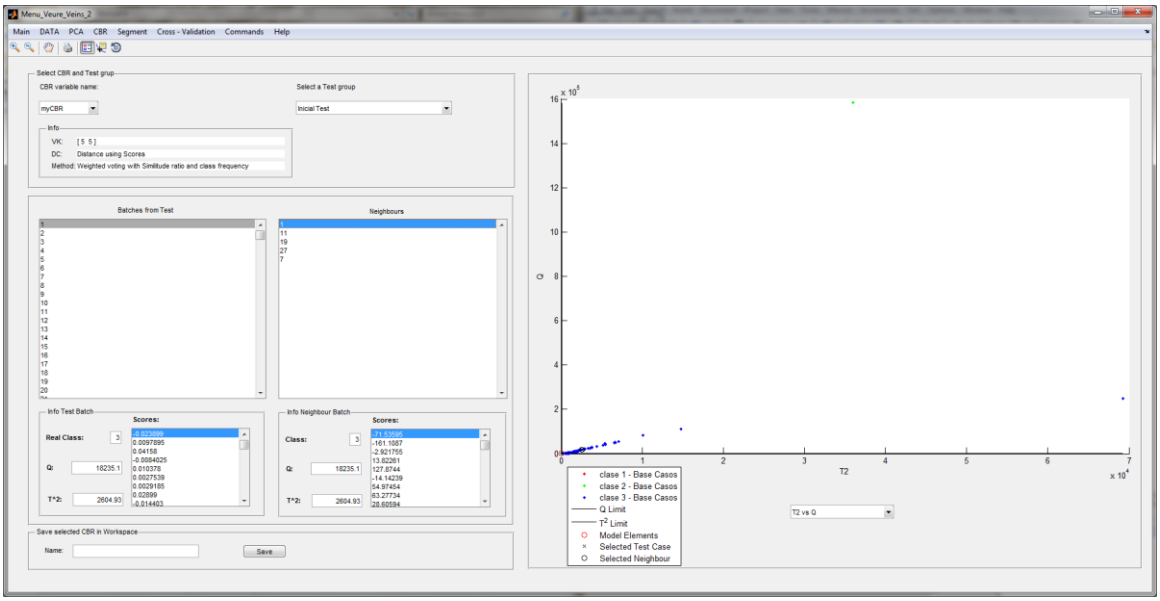


Figura 36

A la Figura 37 podem observar amb més detall el panell de selecció d'objectes. Aquest panell permet escollir la partició (si venim de fer un cross-validation) o bé l'objecte CBR que desitgem (si hem accedit des del menú de CBR). El model PCA associat a aquest CBR(o partició) l'utilitzarem per projectar les dades de test. Tan bon punt s'esculli l'objecte amb el qual volem treballar, ens apareixerà la informació de la configuració que té l'objecte i un selector per escollir l'objecte DATA que utilitzarem coma test.

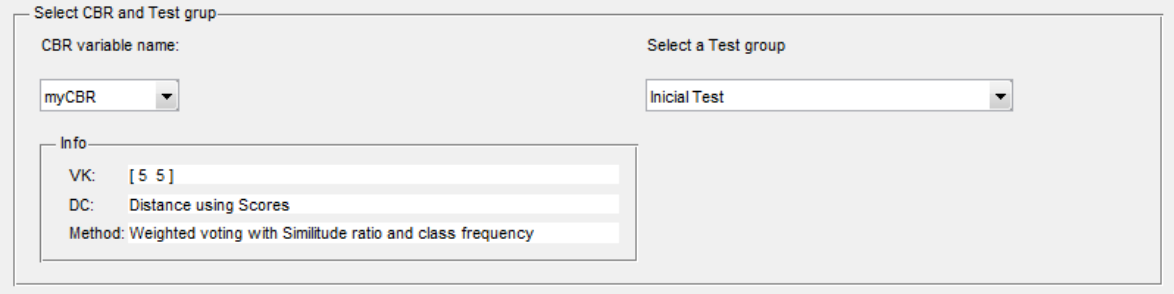


Figura 37

A la Figura 38 podem observar més detalladament el panell on es mostren els lots i els seus veïns.

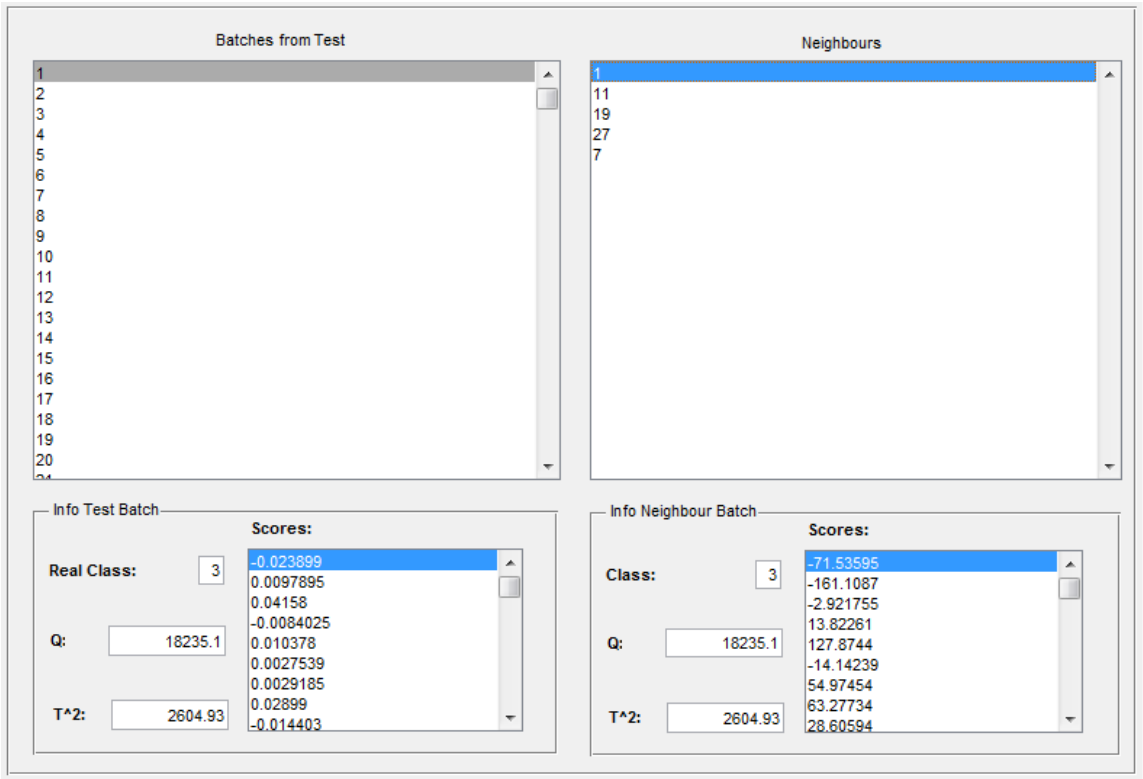


Figura 38

Un cop s'hagin escollit tots els objectes i paràmetres necessaris ja es podrà passar a observar els veïns. A la llista de l'esquerra, en el moment d'escollir un

objecte DATA, apareixeran les referències de tots els seus lots. Per veure'n els veïns només caldrà seleccionar-ne un lot. Aquest fet també ens llistarà la informació de la part de la projecció del lot en qüestió sobre el model PCA. A més a més, sempre que canviem de lot de test, s'actualitzarà la representació en el panell que es mostra a la Figura 39.

A la Figura 39 podem observar més detalladament el panell del Plot.

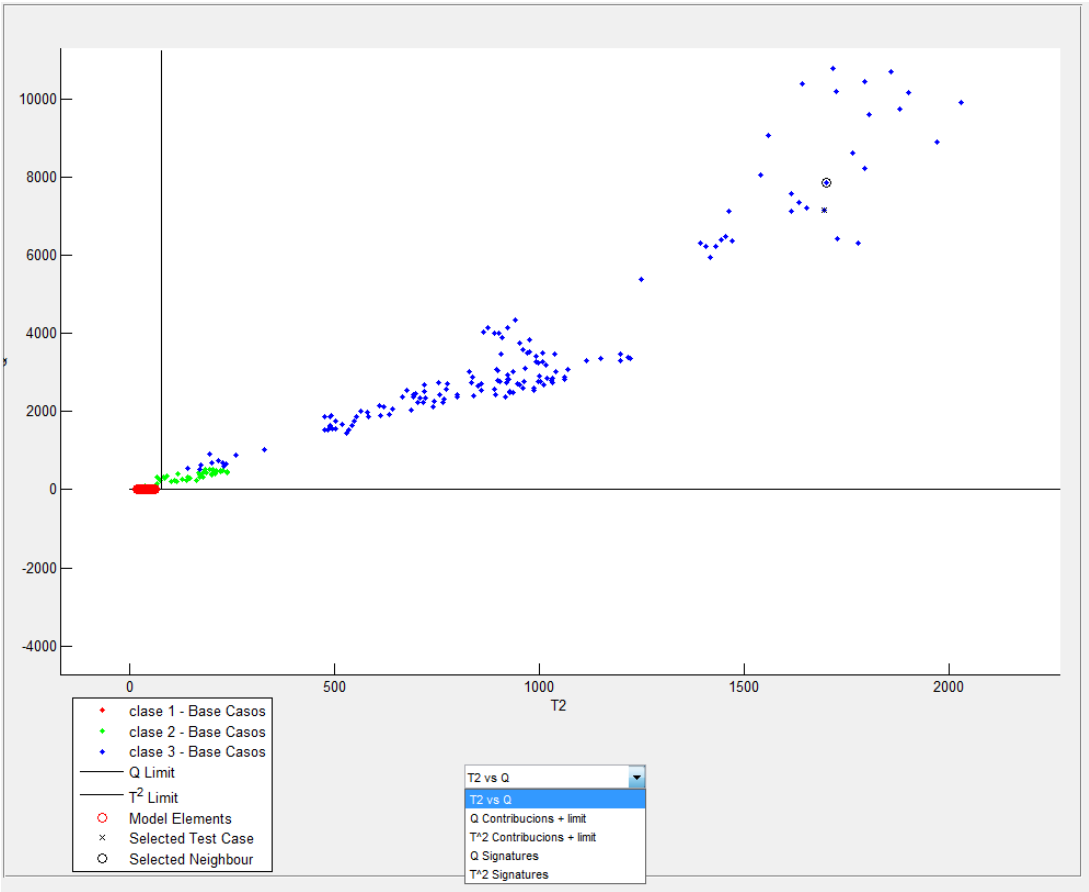


Figura 39

Aquest panell permet seleccionar el tipus de Plot que volem per visualitzar els resultats de forma gràfica. Actualment es permeten 5 tipus diferents de gràfics:

- **T^2 contra Q** - En els gràfics T^2 contra Q, l'eix de les X representa T^2 i l'eix Y representa Q. Tal com indica la llegenda que apareix a la part inferior esquerre els punts de color representen els lots (un color per cada classe) les línies negres representen els límits de T^2 i Q. Els cercles de colors indiquen els casos que hi ha al model. La 'X' negre marca la posició del cas de test i el cercle negre el veí seleccionat. Un exemple de l'aspecte general d'aquest gràfic el trobem a la Figura 39.
- **contribucions al límit de Q** - En aquest gràfic es mostren les fases i les variables indicant-ne el nom i marquen les divisions amb línies discontinues negres (sempre que les duracions de les fases, noms de les fases i noms de les variables estiguin inicialitzades). Tal com indica la llegenda es mostraran les contribucions de 2 casos el cas de test i el veí més proper seleccionat. Un exemple de l'aspecte general d'aquest gràfic el trobem a la **Figura 40**.

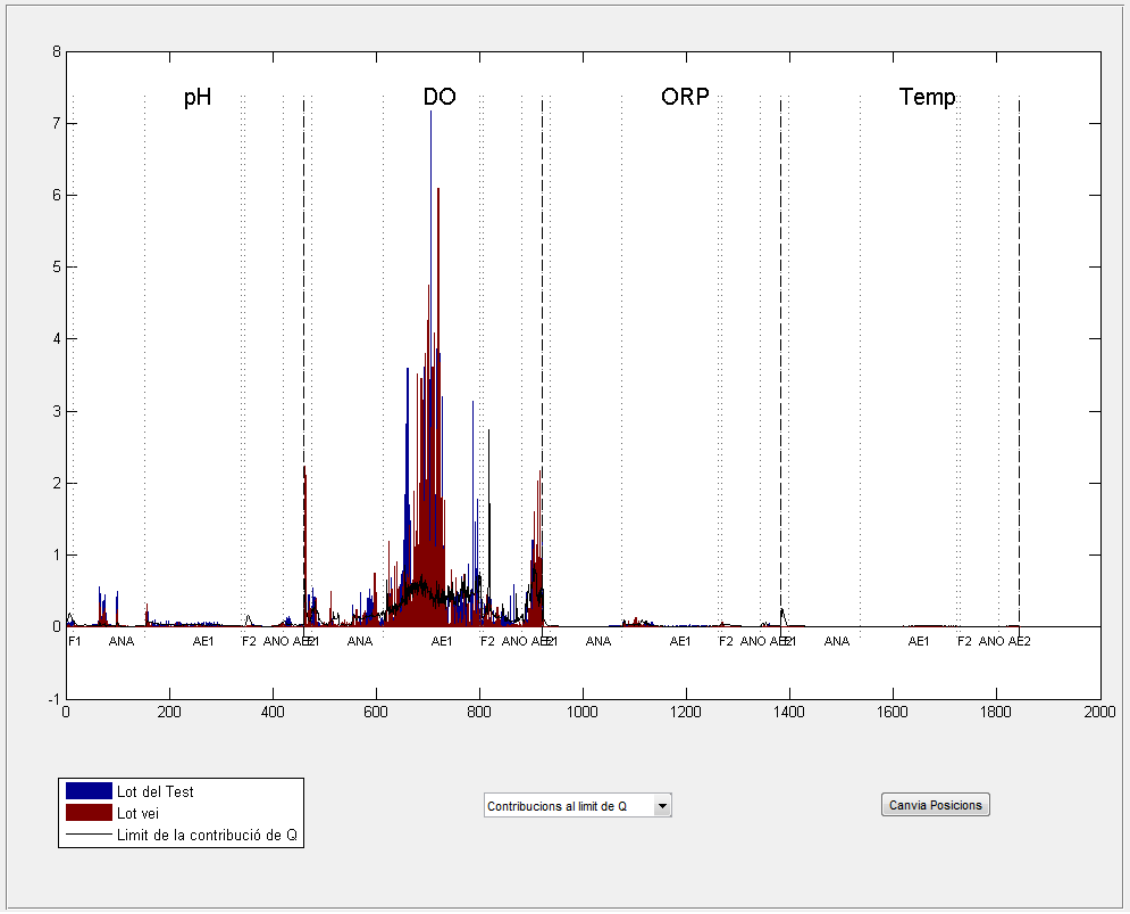


Figura 40

- **contribucions al límit de T^2** - En aquest gràfic es mostren les fases i les variables indicant-ne el nom i marquen les divisions amb línies discontinues negres (sempre que les duracions de les fases, noms de les fases i noms de les variables estiguin inicialitzades). Tal com indica la llegenda es mostraran les contribucions de 2 casos el cas de test i el veí més proper seleccionat. Un exemple de l'aspecte general d'aquest gràfic el trobem a la **Figura 41**.

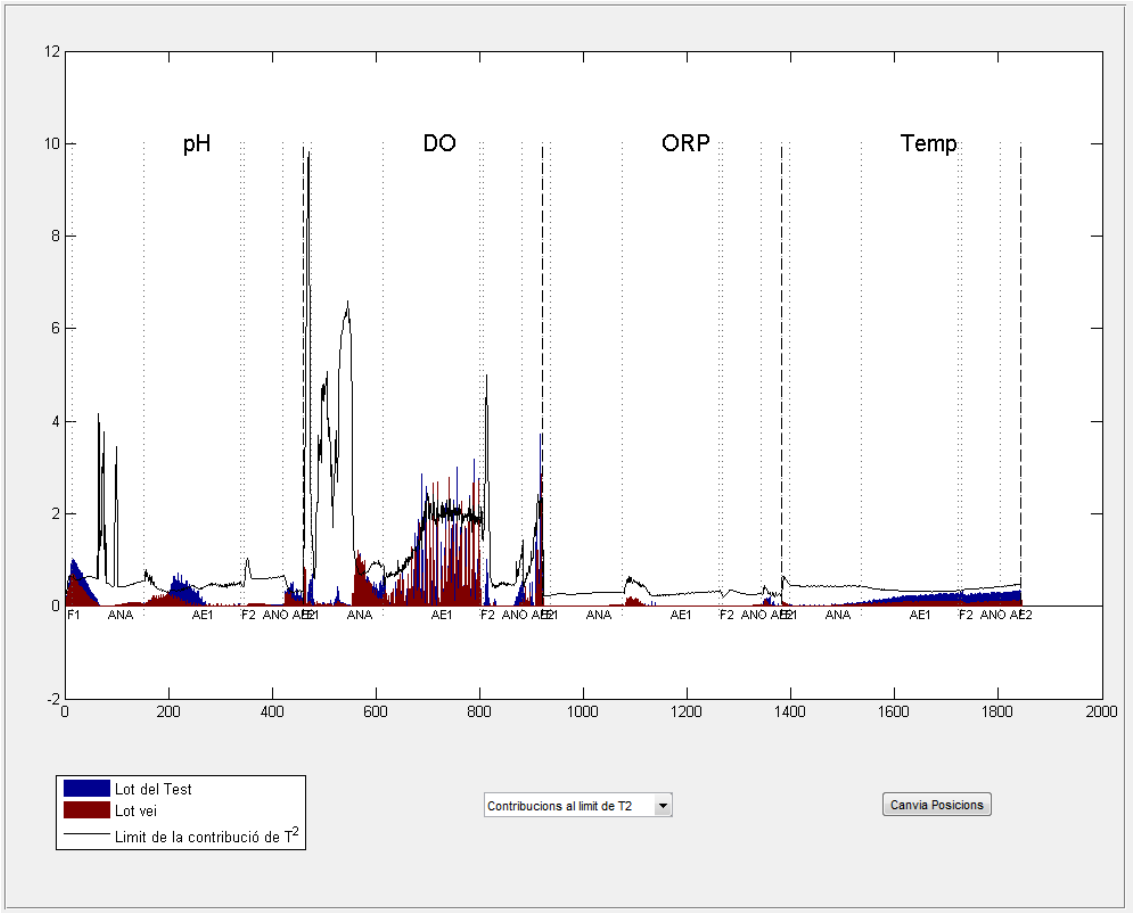


Figura 41

- **Signatures Q** - En aquest gràfic l'eix de les Y hi ha el número de lot

(respecte la posició actual a la Base de Casos), Al llarg de l'eix de les X hi trobarem la signatura del lot corresponent, les zones on falla estaran pintades d'un color que ens indica la classe del lot. Un exemple de l'aspecte general d'aquest gràfic el trobem a la **Figura 42**.

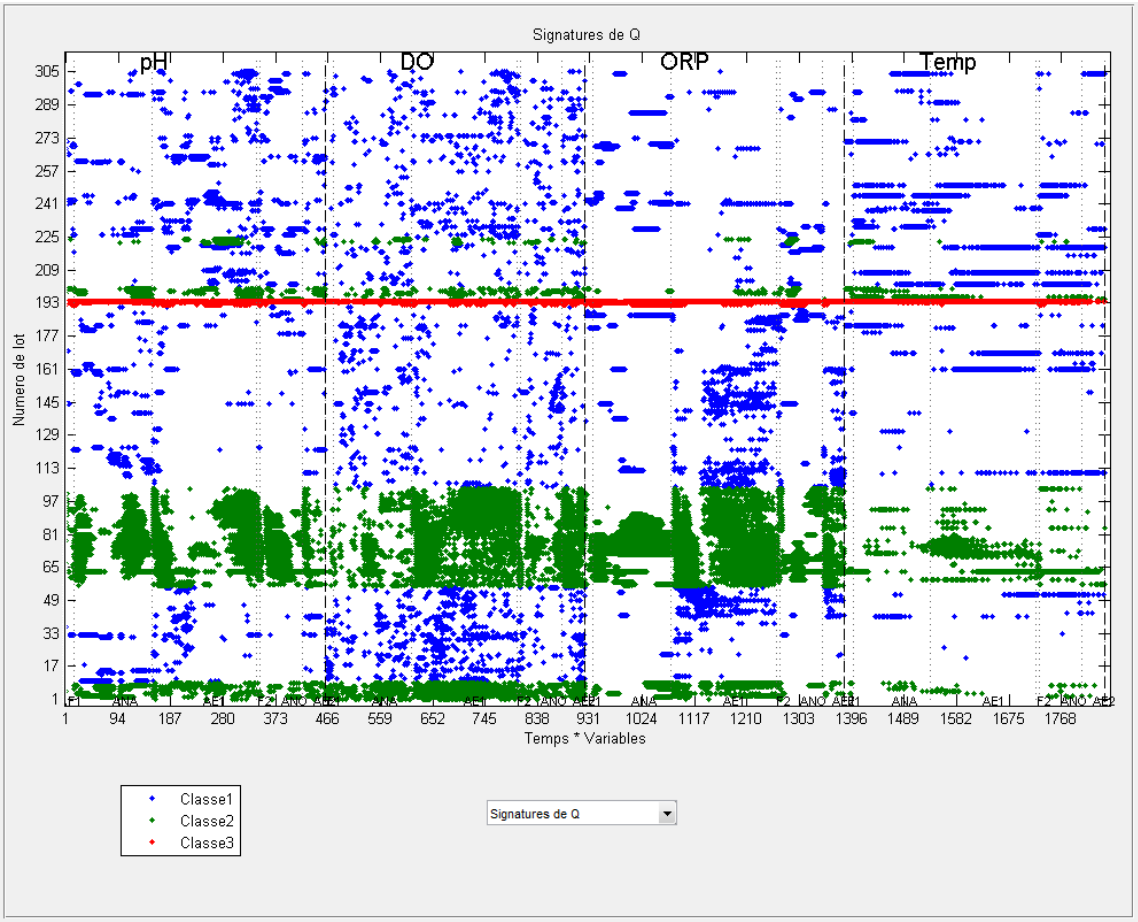


Figura 42

- **Signatures T^2** - En aquest gràfic l'eix de les Y hi ha el número de lot (respecte la posició actual a la Base de Casos), Al llarg de l'eix de les X hi trobarem la signatura del lot corresponent les zones on falla estaran pintades d'un color que ens indica la classe del lot. Un exemple de l'aspecte general d'aquest gràfic el trobem a la **Figura 43**.

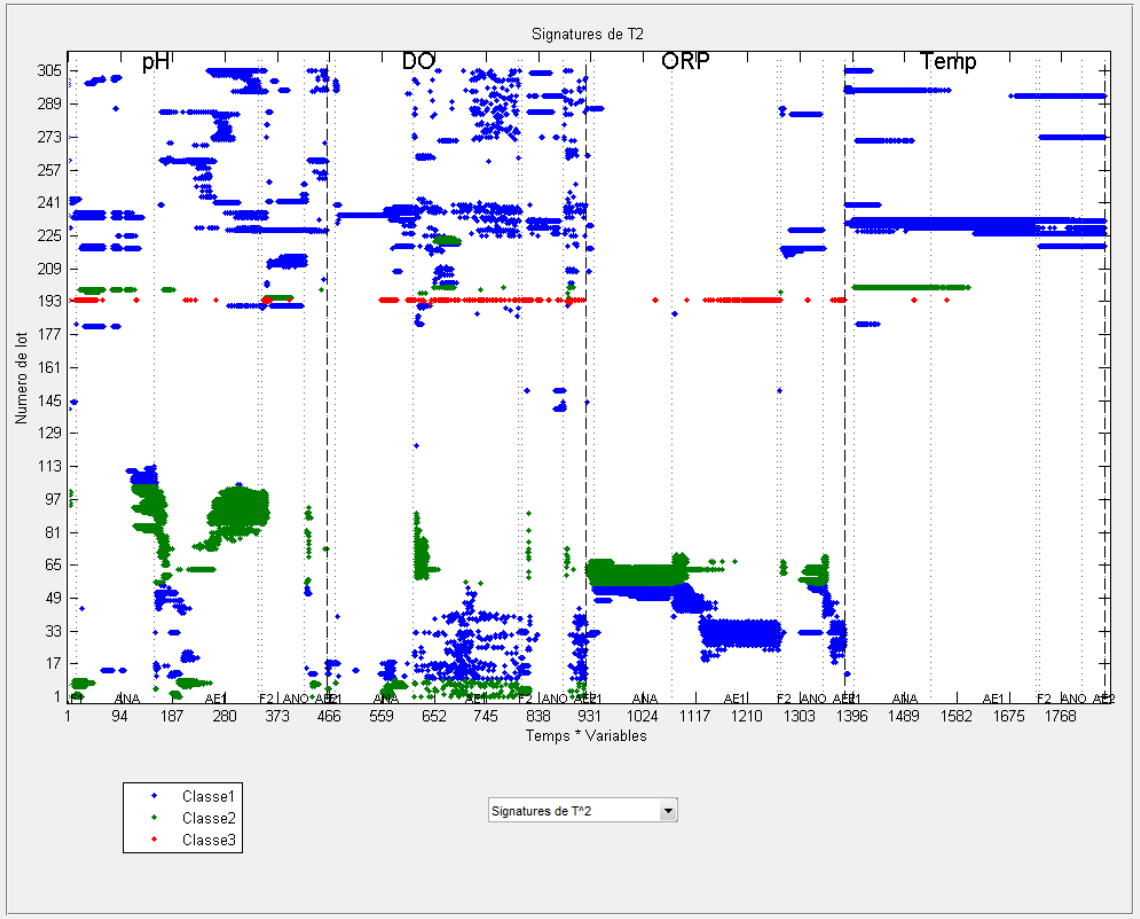


Figura 43

Per tal d'aconseguir sempre una vista òptima s'ha proveït a la finestra d'un menú (**Figura 44**) que disposa d'un seguit d'utilitats per modificar el Plot.

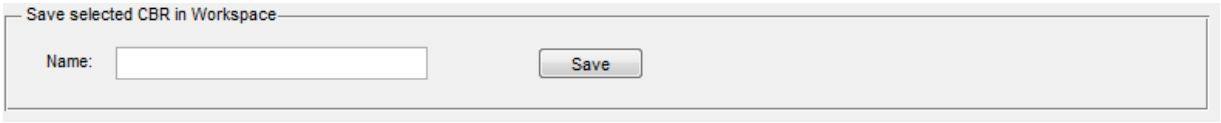


Figura 44

En aquest darrer panell trobarem una utilitat per desar l'objecte CBR que estem utilitzant actualment amb el nom que desitgem. Aquest panell serà especialment útil quan haguem arribat a la pantalla Veure Veïns des de la pantalla Cross Validation ja que si no guardem els objectes que ens interessin els perdrem.

3.7 Disseny de l'exemple d'aplicació

Per aquesta part del PFC les tasques de programació i disseny que s'han hagut de fer no són gens complexes. Per poder fer l'exemple nomes s'ha hagut d'implementar un grup scripts que ens passegessin els fitxers capturats del procés i entressin les dades al Matlab. Fet això tota la feina en aquesta part del PFC consisteix en utilitzar la Toolbox. Per més detalls sobre el format dels fitxers veure el capítol 5.

4 Validació i Avaluació

En aquest apartat detallarem els diferents scripts que s'han fet al llarg del procés d'implementació per comprovar que el codi funciona correctament. Molts d'aquests scripts s'han acabat incloent a mode d'exemples a la interfície d'usuari. Les proves fetes les hem classificat en diversos subapartats depenent de la part del codi que testegen. Aquí no mostrarem totes les bateries de proves que s'han fet ja que són moltes i sovint poc interessants per als usuaris finals. S'ha fet una selecció dels testos més didàctics.

4.1 DATA

Els sets de proves fets per testejar la funcionalitat d'aquesta part del codi són molt simples. La majoria són simples scripts que inicialitzen tots els atributs i comproven que els resultats que retorna l'objecte són els mateixos que s'han entrat.

Aquests testos, degut a la seva simplicitat, s'han acabat incloent a la interfície d'usuari, amb modificacions per fer-los més entenedors, a mode d'exemple didàctic.

Un exemple concret d'aquests scripts per testejar el funcionament de l'objecte DATA és el que comentarem a continuació.

En aquest test utilitzarem les funcionalitats de l'objecte que ens permeten modificar el contingut de l'objecte DATA per comprovar que funcionen correctament.

El primer que es farà és carregar les dues matrius que contenen la informació amb que treballarem:

- **Matriu_Good_Cross_Learning_1_Sort.mat** : Aquest fitxer conté les dades de 59 lots d'un procés de depuració d'aigua en condicions normals de funcionament emmagatzemades en BW. Per cada lot s'han mesurat 4 variables durant 461 instants de temps.

- **DatosProceso.mat** : Aquest fitxer conté informació diversa de cada lot (els noms de les variables mesurades, l'identificador de cada lot, el nom de les diverses fases, etc.).

El codi Matlab que realitza aquesta inicialització es mostra a continuació a la Figura 45.

```
%Netegem el workspace
clear all;
clc;
%Entrem les dades
load 'Matriu_Good_Cross_Learning_1_Sort.mat';
load 'DatosProceso.mat';
```

Figura 45

Seguidament creem un objecte de tipus DATA, utilitzant el constructor per defecte de la classe. Seguidament introduirem les dades de les matrius carregades prèviament a l'objecte.

```
%Creem un objecte DATA buit
dat=DATA();
%Li assignem dades
dat.setRawDataBW(MatriuGCL1sort,4,461,59);
%Al .mat estan en batch-wise
%Posem els noms de les variables del procés
dat.setVariableLabel(model process data.var name);
%Fixem la duració de les fases
dat.setStagesDurations(str2double(model_process_data.phase_time));
```

```
%Donem nom a les fases
dat.setStages(model_process_data.phase_name);
%Assignem les classes. seran totes de la mateixa classe
dat.setClass(ones(1,59));
```

Figura 46

Finalment recuperem les dades de tots els formats possibles.

```
%recuperem les dades en 3D
dadesen3D=dat.getRawData3D();
%recuperem les dades en BW
dadesBW=dat.getRawData(0);
%recuperem les dades en VW
dadesVW=dat.getRawData(1);
```

Figura 47

Utilitzem les funcions per accedir directament als lots.

```
%Recuperem el lot 5 de les dades i la seva classe
[lot classe nom]=dat.getBatch(5);
%Eliminem el lot 5 de les dades
dat.deleteBatch(5);
%Ara el tornem a afegir(sempre s'afegeix al final)
dat.addBatch(lot, classe,nom);
```

Figura 48

Creem una cópia de l'objecte actual.

```
meu = dat.copy();
```

Figura 49

```
%guardem l'objecte 'meu' al fitxer 'meu.txt'

meu.save('meu.txt','csv');

%creem un objecte DATA nou anomenat 'd'
```

```
d=DATA ();

%inicialitzem 'd' amb la informació del fitxer 'meu.txt'

d.load('meu.txt','csv');

%guardem la informació de 'd' a un nou fitxer per comprovar que
%s'ha inicialitzat correctament

d.save('meu2.txt','csv');
```

Figura 50

Podem comprovar, si no hi ha hagut cap error, que els 2 fitxers són iguals i a més a més tant l'objecte 'd', 'meu' com 'dad' són idèntics i amb les mateixes dades que als fitxers.

4.2 PCA

Aquesta classe els testos s'han enfocat més a comprovar que els resultats siguin coherents amb els proporcionats per altres eines com la PLS Toolbox. Aquesta tasca no s'ha pogut automatitzar i s'ha hagut de dur a terme manualment.

Exemple EthCal de la PLS Toolbox implementat amb les llibreries pròpies. Per poder executar aquest exemple és necessari tenir instal·lada la PLS Toolbox per poder accedir a les dades dels exemples.

```
%Exemple EthCal
%Comencem eliminant totes les finestres obertes
close all;

%Carreguem les dades de EthCal
load etchdata;

% Posem les dades en un format més còmode de treballar
```

```
dad = EtchCal.data;
classes = EtchCal.class{3};

%Creem i inicialitzem l'objecte DATA
meu = DATA();
meu.setRawData(dad);
meu.setClass(classes);

%Creem el model PCA amb l'objecte DATA creat
p = PCA(meu);
p.setNormalizationMethod(4);%Escollim el block Scaling com a la Toolbox
%block de l'exemple de la Toolbox són:
%[ones(1,12) * 80]
p.setPCSelectionMethod(0);% Els seleccionarem manualment
%numero components de l'exemple -> 3

p.setLimitMethod(0);% no ho utilitzarem
p.setTrajectoryRemoval(1);% Treure forma mitja actiu

p.pca();%obliguem fer els càlculs ara tot i que no és necessari

%Fem la figura dels scores per poder-la comparar amb la del manual
figure;
p.plotScores(1,2);

%preparem les dades pel conjunt de test (el que projectarem al nostre
% model)
dad = EtchTest.data;
classes = EtchTest.class{3};

%Creem l'objecte DATA que conte els casos de test
test = DATA();
test.setRawData(dad);
test.setClass(classes);

%projectem els casos de test
testprojectat = p.project(test);
```

```
%fem els mateixos plots que l'exemple de la Toolbox
figure;
testprojectat.plotScores(1,2);

figure;
class = testprojectat.getClass;
col=hsv(max(class));%el color que pintarem cada classe
hold on
Q = testprojectat.getQ;
for i=1:max(class)
    index = class == i;
    plot(find(index), Q(index),'.','color',col(i,:));
end
```

Figura 51

Aquest script ens permet comprovar bon funcionament de diverses funcionalitats del nostre objecte PCA mitjançant les dades ofertes per una eina externa com és la PLS Toolbox. Reproduint els plots i obtenint els mateixos resultats.

4.3 CBR

Com passa en els testos de l'Objecte DATA per testejar el funcionament de CBR no disposem de testos molt complexes, els que tenim simplement es limiten a inicialitzar els atributs i a invocar els mètodes més rellevants i delicats.

El primer que farem per testejar la classe CBR és crear-ne una instància. Per fer-ho, podem copiar-ne una de ja existent o bé, podem crear-ne una de nova. Concretament el que fem és crear una instància (amb tots els atributs per defecte) amb el constructor de la classe CBR i després en fem una còpia. Primerament s'ha de crear un objecte PCA, que serà el model del nostre CBR, com es mostra en l'exemple de la Figura 52.

(En el cas d'aquest exemple, disposem de 4 variables amb 424 mostres preses de 70 lots, als quals els hi assignem la classe "1" que a partir d'ara serà per els bons)

```
%carreguem  les dades
load Matriu_LEQUIA_Bons_2A_Final.mat;
load Matriu2A_Bons_3e.mat;
dat=DATA();%creem un nou objecte DATA
dat.setRawDataBW(Matriu2ABons,4,424,70);%Entrem les dades en el format correcte
dat.setVariableLabel(Matriu.Vars);%necessitem que tinguin algun nom
cla1=ones(1,70);%creem el vector de classes, tot a 1 perquè són totes bones
dat.setClass(cla1);%afegim les classes a les dades del PCA
%Creem el Model PCA
M=PCA(dat);
```

Figura 52

Seguidament crearem la base de casos, a la Figura 53 es pot veure el codi que crea la base de casos. En el nostre cas posarem totes les dades que tenim en un mateix objecte DATA que utilitzarem per inicialitzar la base de casos, i on utilitzarem la següent relació de classes:

- Classe 1: Observacions en condicions normals de funcionament (bones).
- Classe 2: Observacions que presenten algun tipus de falla (dolentes).
- Classe 3: Observacions que presenten el mateix perfil que les bones, però amb un increment en les mostres (regulars).

```
%carreguem  les dades
load Matriu2A_Dolents.mat;
cla2=ones(1,91)*2;  %creem les classes posem tot a 2 són tots dolents
aux=DATA();
aux.setRawDataBW(Matriu2ADolents,4,424,91); %Entrem les dades en el format
%correcte
```

```
%ajuntem Bons i dolents
dade=cat(3,aux.getRawData3D(), dat.getRawData3D());
claa=cat(2,cla2,cla1);

load 'Matriu_LEQUIA_Regulars_2A.mat';
dadess = Matriu.Variables;
cla3=ones(1,62)*3;  %creem la classe 3 per ela regulars
%Ajuntem també els regulars
dade=cat(3,dade, Matriu.Variables);
claa=cat(2,claa,cla3);
dat1=DATA();
%creem un objecte dades
dat1.setRawData(dade);%li  afegim les dades
dat1.setVariableLabel(Matriu.Vars); %posem noms a les variables
dat1.setClass(claa);  %afegim les classes
```

Figura 53

Finalment un cop creats el model PCA "M" i la base de casos "dat1" només resta crear el nostre objecte CBR, com es mostra en a la Figura 54.

```
MY_CBR = CBR(M, dat1);
```

Figura 54

Arribats a aquest punt ja podem invocar els mètodes CBR::retrieve, CBR::reuse, CBR::revise , CBR::retain o optimitzar la base de casos amb mètodes com CBR::Drop4, CBR::IB3... com fem a la Figura 55 en forma completa i a la Figura 56 de forma abreujada.

```
vk = [5 3];
dc = 10;
methode = 3;
MY_CBR.retrieve(prov,vk,dc);
MY_CBR.reuse(method);
```

```
MY_CBR.IB2(vk,dc,methode);
MY_CBR.IB3(vk,dc,methode);
MY_CBR.IBUdG(nelem,descartats,vk,dc,methode);
MY_CBR.DROP4(vk,dc,methode);
```

Figura 55

```
MY_CBR.retrieve(prov);
MY_CBR.reuse();
MY_CBR.IB2();
MY_CBR.IB3();
MY_CBR.IBUdG(nelem,descartats);
MY_CBR.DROP4();
```

Figura 56

Mitjançant aquest script podem comprovar de manera ràpida si una distància o una configuració concreta de CBR presenta algun error.

4.4 Accés a bases de dades

La part de connexió del Matlab amb MySQL es va decidir deixar una mica apartada perquè es va veure que no era tan útil com inicialment es pensava. Tot i això tenim un seguit de proves on es testeja l'accés a una Base de Dades MySQL des d'un script de Matlab.

Per tal de fer la comunicació de Matlab amb una base de dades MySQL utilitzarem una Toolbox externa. Existeixen diverses solucions, però la més simple i a l'hora de codi lliure és el projecte: *MySQL Wrapper for Matlab* més conegut com (mym).

La pagina web d'aquest projecte és:
http://sourceforge.net/projects/mym/

Exemple que es va fer per comprovar que la Toolbox funcionava correctament:

```
%fem el login a la base de dades
%(en el nostre cas tenim instal·lada localment si no canviar localhost
%per l'adreça o adreça:port corresponents i podem accedir indicant:
%l'usuari: root (per nostre server local) i de pass:admin (per nostre
%server local))

try
    a = mym('open','localhost','root','admin');
catch err
        errordlg(err.message,'Error');
end
%en cas de no poder fer el login retorna error no hauria d'executar-se
%mai el següent codi però per si decés comprovem!
if ~isempty(a)
    %la Connexió ha estat correcte

    %seleccionem la base de dades amb la qual volem treballar
    resultat = mym('use mydatabase');
    %comprovem el resultat
    if resultat ~= 1
        %no esta creada la base de dades mydatabase
        mym('CREATE DATABASE mydatabase');%la creo
        mym('use mydatabase');%la utilitzem
    end

    %creem una taula anomenada STATION amb 5 atributs:
    % ID , KEY, CITY, LAT_N i LONG_W

    try
        [~]=mym('CREATE TABLE STATION (ID INTEGER PRIMARY KEY, CITY
CHAR(20), STATE CHAR(2), LAT_N REAL, LONG_W REAL);');
    catch err
        %la taula ja existia!
        errordlg(err.message,'Error');
    end
    %afegim dades
    try
        [~]=mym('INSERT INTO STATION VALUES (13, ''Phoenix'', ''AZ'',
33, 112);');
    catch err
        errordlg(err.message,'Error');
    end

    try
        [~]=mym('INSERT INTO STATION VALUES (44, ''Denver'', ''CO'',
40, 105);');
    catch err
        errordlg(err.message,'Error');
    end

    try
        [~]=mym('INSERT INTO STATION VALUES (66, ''Caribou'', ''ME'',
```

```
47, 68);');
    catch err
        errordlg(err.message,'Error');
    end

    %consultem la taula Station
    try
    out= mym('SELECT * FROM STATION;');%retorna tants atributs com
columnes demanem!!
    catch err
        errordlg(err.message,'Error');
    end

    %finalment eliminem la taula
    try
    [~]=mym('DROP Table STATION;');
    catch err
        errordlg(err.message,'Error');
    end

    %tanquem la conecció!
    mym('closeall');
end
```

Figura 57

Aquesta Toolbox s'encarrega de definir les funcions necessàries per accedir des de Matlab a una base de dades MySQL.

4.5 Cross-Validation

Aquest és un dels testos més simples que tenim, és un mètode que comprova el funcionament del Cross-Validation amb totes les configuracions possibles de CBR pel que fa a distància, voting i optimització. El codi d'aquest script el podem observar a la Figura 58.

```
%borrem tots els objectes que creem
clear
clc
close all;

%Creem un objecte DATA carregant-lo des de fitxer
C_B=DATA();
C_B.load('C_B2.txt','csv')
```

```
%Configuracions del CBR
vk=[5 3];
divs = 5;

%Configuracions de PCA
W = 0;%Unfolding Method
est =1;%Normalization Method
FM = 0;%Trajectory Removal
Metode = 1;%P.C. Selection Method
limits =0;%Limit Method
conts =1;%Contribution Method
classes = unique(C_B.getClass);
parami = 0.5;%Paràmetre per les distancies  17 i 18

%Comencem a provar totes les distancies amb aquesta configuració de PCA
for method=1:3 %voting
    for net = 1:5 %optimització
        for dc=1:18 %distancia
            try
                %Fem un cross-validation amb aquesta configuració!

Cross4(C_B,divs,vk,dc,method,net,W,est,FM,Metode,limits,conts,classes,pa
rami);
            catch err
                %si hi ha error mostrem el missatge i la configuració
                % que l'ha provocat

                dc
                method
                net
                err
            end

            %tanquem totes les finestres!
            close all;
        end
    end
end
```

Figura 58

Aquest test no només ens ha servit per testejar el funcionament global de l'aplicació, ja que en fer un Cross-Validation s'utilitza CBR, PCA i DATA, aquest test el vam utilitzar mitjançant la utilitat Code inspector del Matlab per saber en quines parts del codi s'estava consumint més temps de còmput, quantes vegades s'invocava cada funció,... Aquesta informació proporcionada pel Code inspector va ser molt útil per a optimitzar el codi i es van poder detectar diversos punts on es feien càlculs innecessaris o que el codi no estava prou ben optimitzat.

4.6 Interfícies

Les interfícies no permeten ser testejades mitjançant scripts. Per tant ha estat necessari provar el codi manualment finestra a finestra i explorant tants casos com fos possible. Per testejar les interfícies ens ha ajudat molt l'exemple d'aplicació el qual ha servit a més de exemple d'ús com a test.

4.7 Avaluació del rendiment

A continuació detallarem les utilitats que hem utilitzat per testejar el rendiment del codi implementat. Exposarem també com s'ha estimat el tamany màxim de les dades i la relació existent entre el temps de càlcul i el tamany de les dades.

4.7.1 Eines utilitzades

Per tal d'optimitzar el codi Matlab disposa de diverses eines. Nosaltres les hem utilitzat per trobar els punts on el codi necessitava optimitzacions.

4.7.1.1 Profiler

Una de les utilitats més interessant ha estat el Profiler. Aquesta Aplicació s'encarregarà de recollir els temps d'execució, temps en espera, nombre de crides a cada funció,... fetes mentre el profiler sigui actiu. Aquests resultats donen fàcilment una idea de quins són els punts crítics del nostre codi i a on cal fer optimitzacions.

4.7.1.2 Tic;Toc

Aquestes 2 comandes són la manera més simple de saber el temps que triga un tros de codi en executar-se. La manera d'utilitzar aquestes comandes és molt

simple, la comanda tic inicia un timer, la comanda toc mostra per consola el valor del timer.

Amb aquestes 2 eines podrem trobar fàcilment el tems que triga en executar-se una porció concreta del nostre codi.

4.7.2 Resulltats de rendiment

Tots els testos realitzats a continuació s'han fet sobre dades aleatòries. I el seu objectiu és comprendre com es comporta la Toolbox respecte el volum de dades rebudes.

Velocitat de creació del Model PCA

En aquest cas s'ha calculat el temps que triga la Toolbox en calcular un model donat un objecte DATA d'on en variarem el nombre de lots, variables i instants de temps.

Lots = 100
Instants de temps = 100
Les variables les variem de 1 a 100 per veure l'evolució del temps de còmput. El resultat utilitzant un desdoblat Batch Wise el trobarem a l'esquerra Figura 59 de la i utilitzant Variable Wise a la dreta.

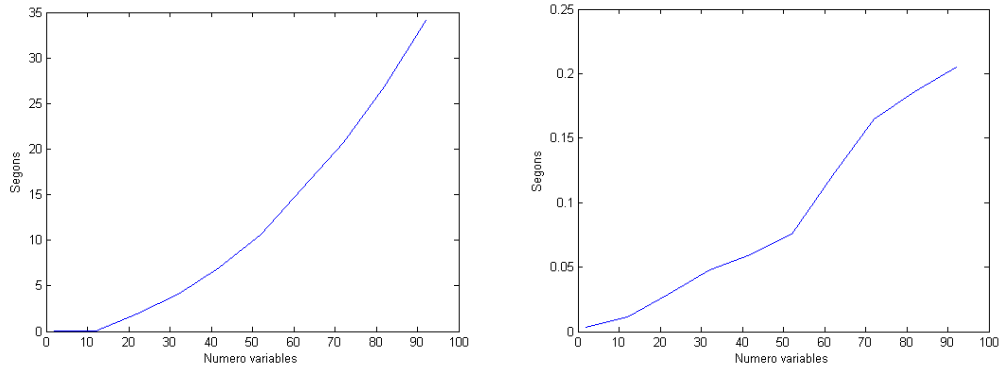


Figura 59

Com podem observar a la Figura 59 depenent del tipus de desdoblat que s'utilitza el temps de creació del model varia sobretot degut al temps de còmput de les components principals.

Velocitat de neteja de la base de casos

En aquest test comprovarem el temps que triga l'aplicació en realitzar una neteja de la base de casos depenent del nombre de casos que tenim. Compararem els temps que triga cada un dels algorismes, IB2, IB3, DROP4 i IBUdG.

A la Figura 60 hi tenim representats el temps que triga en fer una neteja cada un dels 4 algorismes. Les proves s'han fet amb dades de les següents característiques:
Lots variarem de 100 a 10100 augmentant de 1000 en 1000.
Instants de temps 100
Variables 10

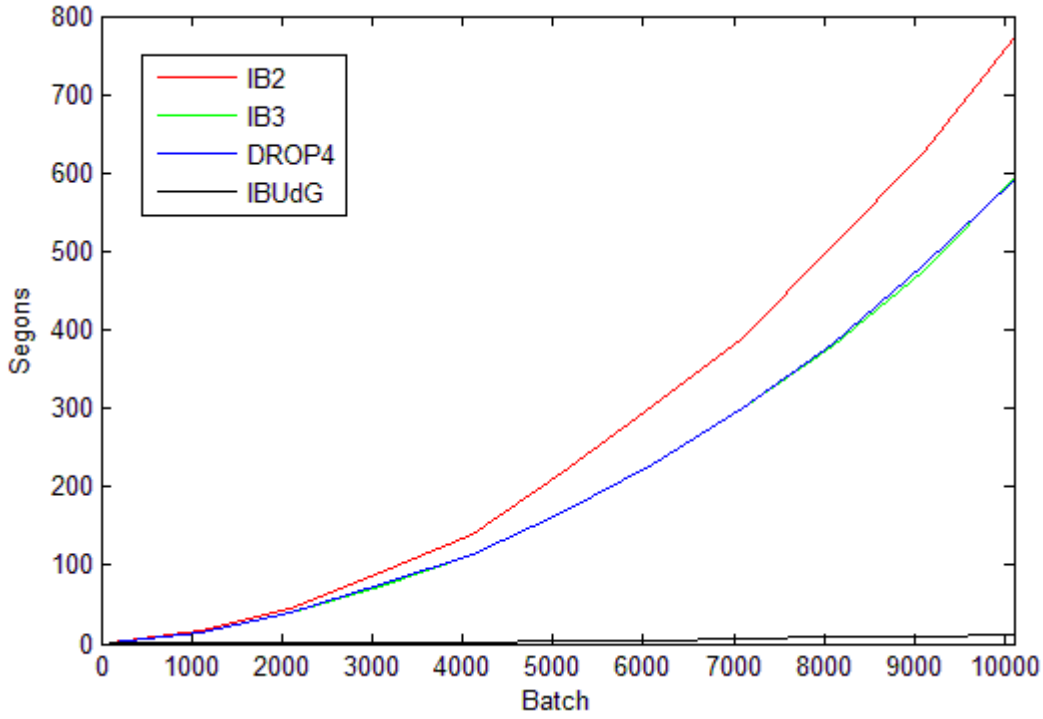


Figura 60

Com era d'esperar IB2, IB3 i DROP4 són clarament quadràtics respecte el nombre de lots. Tot i les optimitzacions no hem aconseguit millorar aquest rendiment. Concretament les proves es fan amb una base de casos on les classes són completament solapades correspon al pitjor cas en quan a temps de còmput. Si les classes no fossin solapades observaríem que Drop4 és més lent que IB3 i IB2 ja que aquests s'acosten al comportament lineal quan les classes són disjuntes. Pel que fa al mètode IB_UdG és de cost lineal respecte el nombre de casos, mitjançant operacions matricials per les quals el Matlab esta mol optimitzat hem aconseguit apropar-nos molt al cost constant. En tots els algorismes hi ha inclosos el temps de còmput de les distancies.

Limitacions de la Toolbox

Matlab esta creat amb java, per tant estarem limitats als tamanys de variables que permeti java en cada sistema. Si executem la comanda memory ens donarà els valors màxims de memòria que es poden utilitzar a la nostra maquina (Ram, Ram + swap,...). En Matlab un double ocupa 8bytes. Per tant una matriu de 10.000x10.000 doubles ocuparà uns 760MB de memòria. Tot i que es possible és molt recomanable no superar mai el valor de la ram del sistema perquè això faria que el sistema fos extremadament lent.

Per experiències pròpies a la nostra Toolbox el punt crític en memòria és el moment de calcular les components principals. Per calcular les components principals sobretot en Batch Wise es crearan matrius molt grans concretament de (variables*temps)X(variables*temps) de doubles, aquest valor es fàcilment un valor gegantí 100 variables i 100 instants de temps ja són 763Mb i si fem els càlculs per 100 variables i 1000 instants de temps són 74,5GB que fa que els càlculs siguin impossibles de fer al nostre sistema. En cas d'intentar càlculs que necessiten mes memòria de la que disposem ens apareixerà un error indicant "java out of memory" i no es permetrà fer els càlculs.

5 Exemple d'aplicació

Com a exemple d'aplicació real es va decidir utilitzar el conjunt de llibreries i Interfícies que s'havien creat en la detecció de l'estat en un procés real de depuració d'aigües. Aquesta aplicació ajuda a més de testejar l'aplicació a demostrar-ne una de les possibles aplicacions reals.

5.1 Introducció i motivacions

El tractament d'aigües ha esdevingut en els darrers anys un requeriment indispensable per tal de mantenir el cicle natural de l'aigua net de residus. El creixement continu dels nuclis de població ha provocat que les plantes depuradores d'aigües residuals hagin de tractar cabals més elevats amb les mateixes instal·lacions construïdes anys enrere. Per tal de solucionar els problemes d'espai que presenten les estacions depuradores d'aigües residuals s'ha estat estudiant la possibilitat d'utilitzar els reactors discontinus seqüencials (acrònim en angles, SBR) . Aquestes plantes tenen l'avantatge d'ocupar molt menys espai físic per tractar el mateix cabal d'aigua que la tecnologia utilitzada actualment.

En l'ús de SBR actualment s'està investigant en diversos camps. Un d'ells és l'ús de fang granular aerobi. Aquests fangs presenten avantatges respecte els fangs actius utilitzats actualment, entre ells la possibilitat de tractar més cabal d'aigua que els fangs actius.

Els processos que utilitzen el fang granular aerobi presenten dos punts de treball diferenciats. El procés pot canviar d'estat depenent de les matèries primes que es tracten amb el procés. Els estats d'un procés de depuració d'aigües amb fang granular aerobi depenent de la mida de les partícules del grànuls de fang.

Per contra, el procés de granulació (canvi d'estat) en la utilització de fang granular aerobi té encara molts buits de coneixement que cal investigar. El procés de granulació es basa en un procés de 2 passos similar al creixement de cristalls. Aquest procés consisteix en introduir partícules fines en el procés, anomenades portadors, en la superfície d'aquests portadors el bacteris es desenvolupen fins a esdevenir grànuls madurs.

Un dels punts claus en la utilització de fang granular aerobi és la detecció de la granularitat del procés (estat en que es troba). Segons la mida de les partícules del fang es defineixen clarament 2 estats. . La diferencia visual entre els 2 estats la trobem a la Figura 61.

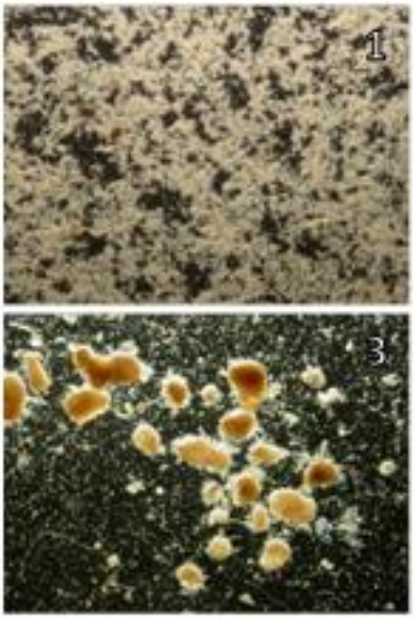


Figura 61

- **Floccular** – Quan la mitja de la mida de les partícules tenen és inferior a 200 µm, imatge superior de la Figura 61.
- **Granular** – Quan la mitja de la mida de les partícules supera els 500 µm, imatge inferior de la Figura 61.

Quan la mitja de la mida de les partícules es troba entre 200 μm i 500 μm el procés està en un estat de transició des d'on fàcilment pot evolucionar cap a qualsevol dels 2 estats.

El nostre exemple d'utilització de les llibreries és centrarà en estudiar les dades que tenim d'un SBR, per tal de detectar-ne l'estat (granulars o be flocular).

5.2 SBR

SBR són les sigles de sequence batch reactor, la característica principal d'aquests reactors és que tot el procés de depuració es produeix en un sol tanc. El procés de depuració d'aigua es divideix en fases. Utilitzant un SBR en totes elles l'aigua residual es mantindrà al mateix tanc, amb altres mètodes de depuració d'aigua és necessari canviar l'aigua de tanc en canviar de fase.

El cicle natural d'un SBR es compon de les següents fases:

1. Emplenat: Es bombeja l'aigua dins el tanc per ser tractada.
2. Reacció: Un seguit de fases aeròbica i anòxica que fan que els bacteris del fang granular actuïn netejant l'aigua.
3. Precipitació: En aquesta fase s'elimina l'excés de fang.
4. Buidat: Es treu l'aigua del tanc deixant-lo preparat per tornar a començar el procés.

5.3 Descripció de les dades

Els fitxers que ens arriben, proporcionats per LEQIA, tenen les dades de 2 SBR. Aquests fitxers són el resultat de les captures fetes per diversos sensors de les plantes potabilitzadores durant la durada del procés. Aquests fitxers disposen de molta informació sobre el procés, si ens fixem amb la capçalera dels fitxers trobem que te sempre el format que es mostra a la Figura 62.

```
Nom del pla de treball:

Número d'etapes:

Set point oxigen1:

Set point temperatura1 (°C):

Set point oxigen2:

Set point temperatura2 (°C):

Bany:

Temps total(min.):

Data:


Hora pH O2 RedOx Temp pHana O2ana RedOxana Tempana aire aire2
```

Figura 62

En el nostre cas ens interessarà el pla de treball i la data perquè són els que ens permetran trobar les fases del procés que es van utilitzar un dia concret i ens permetrà també trobar els resultats de les analítiques corresponents al dia.

Pel que fa a les dades proporcionades pel fitxer, nosaltres nomes necessitarem les corresponents a l'SBR-1 { pH O2 RedOx Temp}. En el nostre cas ens centrarem en l'SBR-1, ja que és on es pot observar el canvi de granularitat que ens interessa localitzar. La resta de dades del fitxer podrem ignorar-les ja que no

corresponen al nostre SBR o bé no tenen cap afecte sobre la granularitat del nostre procés.

Per tal de poder llegir els fitxers de GRASTAC i carregar les dades al Matlab s'ha creat un paquet de funcions que anomenem Lector GRASTAC. A més de llegir els fitxers aquestes funcions també retornen les dades de les diverses analítiques donada una data en concret.

Una dada important per nosaltres serà la granularitat del procés. Les analítiques de mida de partícula no es duen a terme periòdicament, per tant ens haurem d'adaptar a les dades que tenim. També haurem de tenir en compte que al ser un SBR de proves van variant la configuració de les fases per estudiar-ne el comportament.

Si grafiquem totes les dades de granularitat que tenim al llarg del temps obtenim el gràfic representat a la Figura 63.

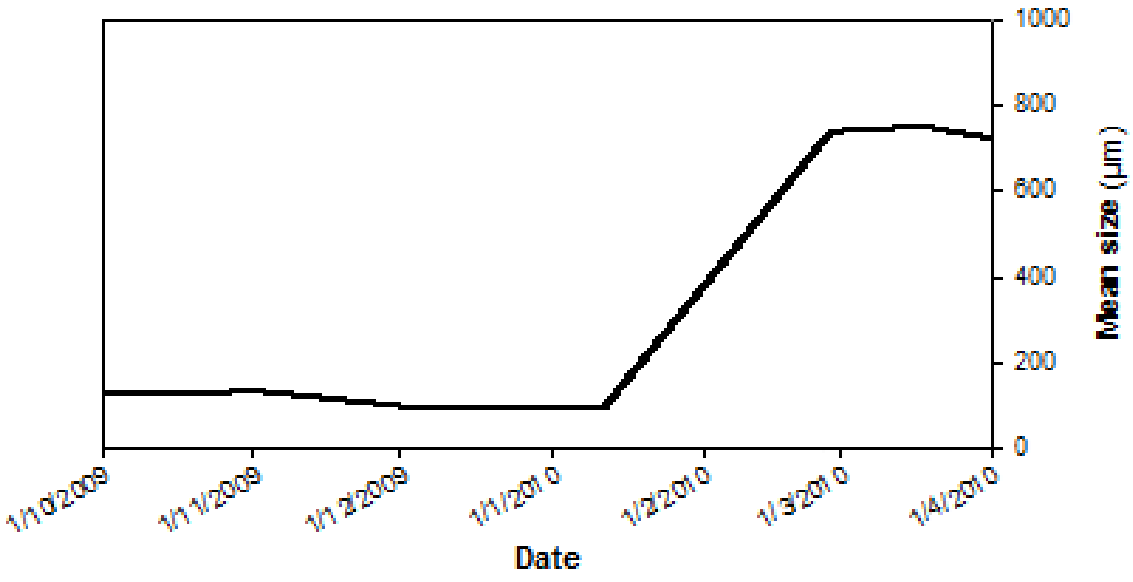


Figura 63

Podem considerar que el procés es troba en un estat granular quan la mitjana de les partícules superen els 500 µm. Com podem observar a la gràfica anterior el canvi de granularitat es produeix entre les observacions dels dies 12/1/2010 i 26/2/2010.

Altres dades importants:

- El sensor de Ph va ser canviat el dia 20/01/2010 perquè fallava des de 11/01/2010.
- La configuració de l'SBR es manté estable de 24/09/2009 fins a 23/03/2010 Anteriorment i posteriorment a aquestes dates la configuració és diferent i per tant a priori no podrem utilitzar dades de fora d'aquest període per construir el model del problema.

5.4 Assignació de classes

Les nostres llibreries treballen nomes amb classes numèriques per tant haurem d'assignar classes als següents estats del procés:

- **Flocular** – Els donarem la classe 1 i seran tots els Lots que sabem segur que són floculars. Basant-nos amb les dades que tenim sobre el procés exposades en l'apartat anterior assignarem la classe 1 a tots els Lots compresos entre 24/9/2009 i 19/12/2009.
- **Granular** – Els donarem la classe 3 i seran tots els Lots que sabem segur que són granulars. Basant-nos amb les dades que tenim sobre el procés exposades en l'apartat anterior assignarem la classe 3 a tots els Lots compresos entre 26/02/2010 i 23/03/2010.
  - **Desconegut** – En realitat no es un estat del procés, és una classe que utilitzarem per marcar tots els lots que no estem segurs de la classe que tenen. Ja sigui perquè el sensor de pH fallava o perquè no tenim

analítiques suficients per ajustar més el dia que es produeix el canvi d'estat. Seran els casos que entrant-los al CBR un cop creat intentarem descobrir-ne la classe real i així saber quan es produeix realment el canvi de granularitat.

Aquesta assignació de classes també la tenim representada visualment a la Figura 64.

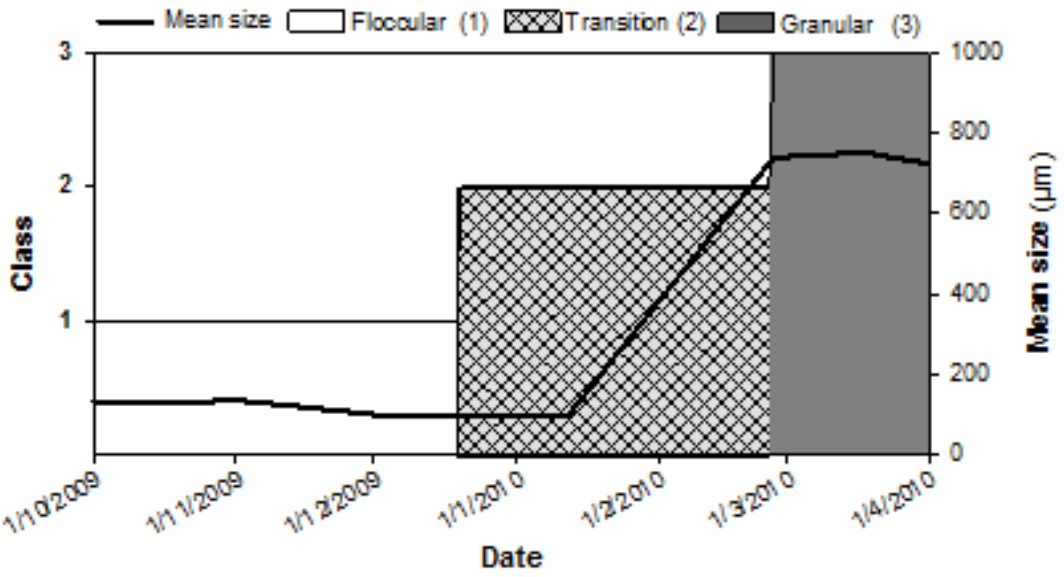


Figura 64

5.5 Procediment

En aquest apartat es descriurà pas per pas que s'ha hagut de fer i com s'ha fet per obtenir els resultats finals. El procediment descrit no és l'únic, i correspondrà a l'usuari escollir la forma que li resulti més practica per arribar als resultats que desitja.

5.6 Adequació de dades

El primer que cal fer és llegir tots els fitxers de dades que ens ha proporcionat LEQUIA. Com ja hem esmentat anteriorment, s'han implementat unes funcions per llegir i carregar aquests fitxers al Matlab, per tant les utilitzarem.
Aquestes funcions les podem trobar a la carpeta Lector GRASTAC. El primer pas doncs si volem utilitzar-les es incloure-les al path del Matlab abans de començar a treballar.

Les dades dels SBR es troben a la carpeta GRASTAC TOT inclosa en el CD. Per començar a llegir totes les dades utilitzarem la funció Readfolders aquesta funció llegirà tots els fitxers de la carpeta que li indiquem per paràmetre i ens crearà una estructura de dades al workspace amb la qual treballaran la resta de funcions proporcionades a la carpeta Lector GRASTAC.

Per tal de poder utilitzar les interfícies gràfiques i funcions de les llibreries el que haurem de fer és crear un objecte DATA amb les dades que ens proporcionen les funcions del Lector GRASTAC.

Per facilitar la creació de l'objecte DATA s'ha inclòs en la carpeta Lector GRASTAC un script anomenat crear_var_dad que en executar-lo crearà al workspace un objecte amb totes les dades assignades de manera automàtica.

Un cop creat l'objecte DATA que l'script crea i anomena dad ja podem començar a utilitzar les interfícies i/o comandes per treballar amb les dades de GRASTAC.

5.7 Modelat del procés

Un cop hem carregat les dades al Matlab i hem creat un objecte DATA el següent pas és trobar una configuració acceptable per al model PCA i per el CBR.

Se'ns plantegen diverses configuracions possibles. Tant per PCA com per CBR però en el nostre cas ens centrarem més en trobar una distància que ens permeti diferenciar al millor possible les classes 1 i 3 que no en trobar una configuració optima de PCA per estalviar temps.

Hi ha moltes maneres de trobar la configuració optima utilitzant la Toolbox. Però possiblement la manera més fàcil i ràpida de provar configuracions diferents i obtenir informació de lo bona que és la configuració és utilitzar la pantalla per fer Cross-Validations. Per utilitzar aquesta pantalla només ens fa falta l'objecte DATA amb totes les classes assignades que hem creat anteriorment.

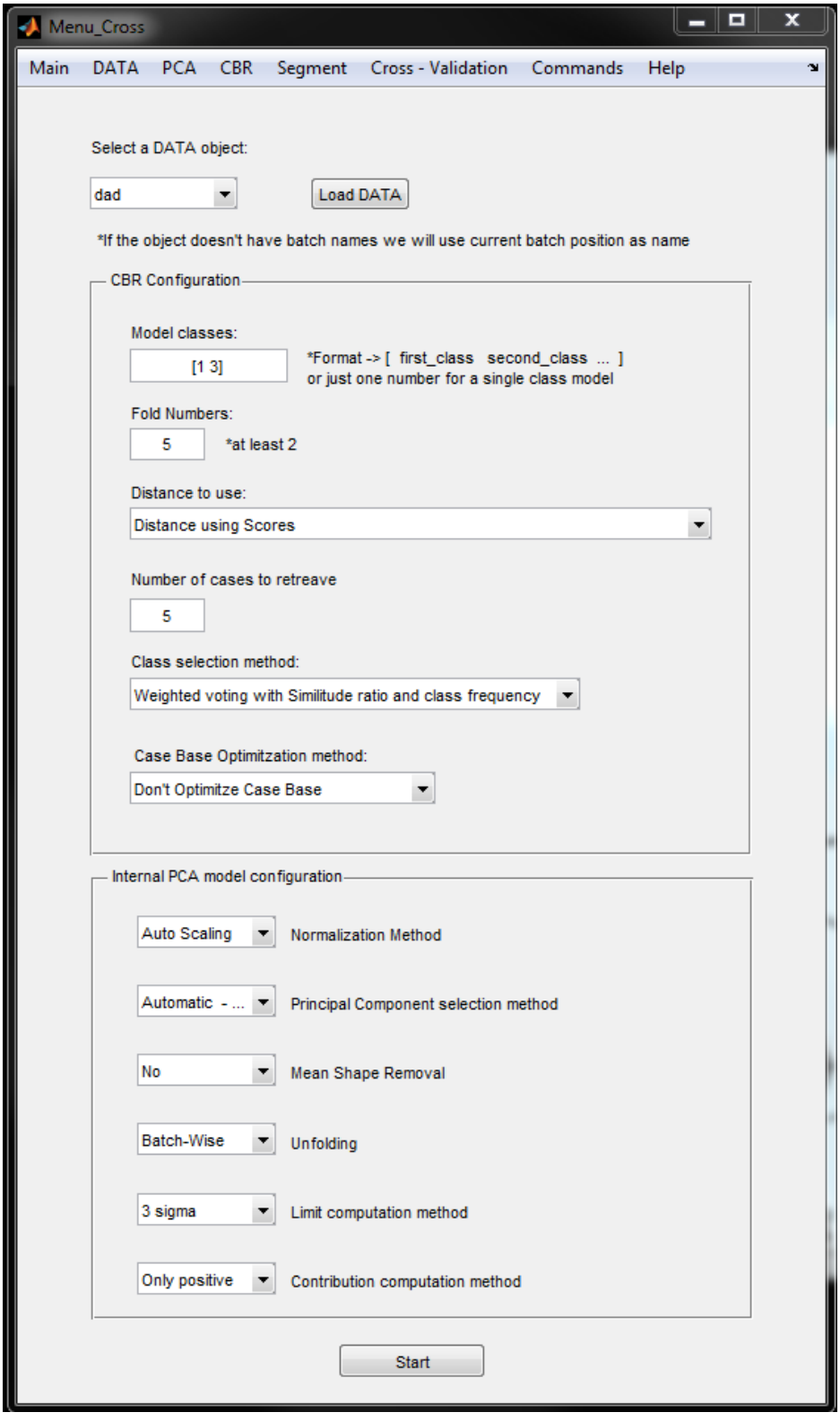

Menu_Cross
Main
DATA
PCA
CBR
Segment
Cross - Validation
Commands
Help
Select a DATA object:
dad
Load DATA
*If the object doesn't have batch names we will use current batch position as name
CBR Configuration
Model classes:
[1 3]
*Format -> [first_class second_class ...]
or just one number for a single class model
Fold Numbers:
5
*at least 2
Distance to use:
Distance using Scores
Number of cases to retreave
5
Class selection method:
Weighted voting with Similitude ratio and class frequency
Case Base Optimitzation method:
Don't Optimitze Case Base
Internal PCA model configuration
Auto Scaling
Normalization Method
Automatic - ...
Principal Component selection method
No
Mean Shape Removal
Batch-Wise
Unfolding
3 sigma
Limit computation method
Only positive
Contribution computation method
Start

Figura 65

En la pantalla de Cross-validation el que farem és provar configuracions de PCA/CBR. En el nostre cas primer de tot estudiarem la possibilitat de realitzar el Model amb la classe 1, la classe 3 o bé amb les 2 a l'hora. A la pantalla indicarem a Model classes les classes que volem utilitzar per fer el model 1, 3 o be [1 3] per utilitzar les 2 classes.

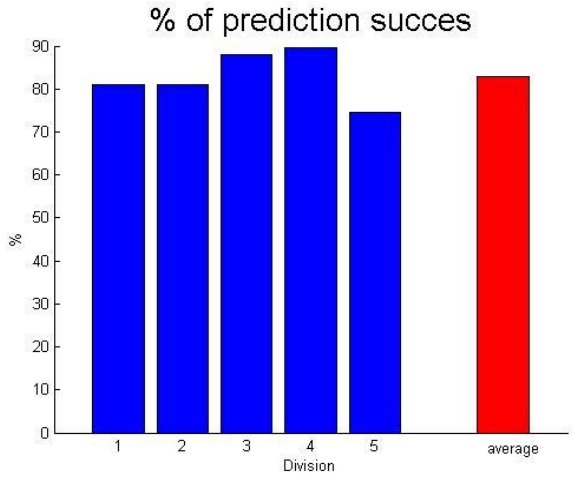
Realitzar el model amb una sola classe farà que es busquin correlacions dins les variables quan el procés es troba en aquest estat. En projectar sobre el model creat només amb dades d'un estat dades d'un altre estat del procés s'observaran (si els estats són prou diferents) valors inusuals de Q (Errors de correlació). En aquest cas ens serà interesant sobretot provar les distàncies de CBR basades en l'espai residual ja que és aquí on teòricament s'hauran de reflectir les diferencies entre els estats del procés.

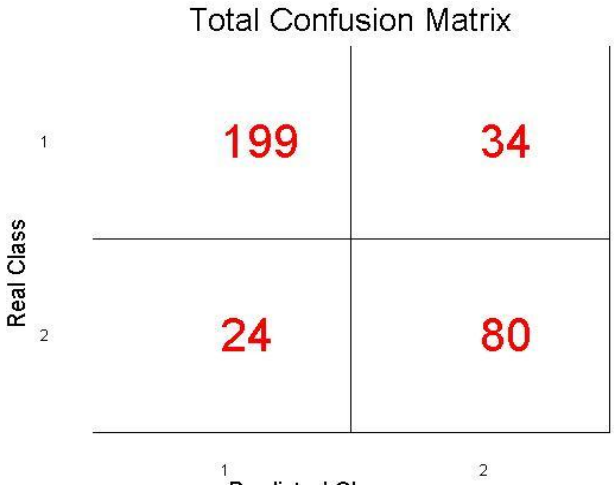
Per altre banda podem també crear el model utilitzant ambdós estats del procés (flocular i granular). Utilitzant aquesta estratègia PCA es focalitzarà en distingir d'entre els estats (la variabilitat entre classes és major que la variabilitat dins d'una mateixa classe). Degut a aquesta peculiaritat prendran especial significat les distàncies de CBR basades en l'espai de projecció (scores i T2).

L'altre paràmetre que variarem, en el nostre cas és la distancia que utilitza CBR per medir la distancia entre els casos. Cada distancia es fixarà en uns paràmetres de control o un altres segons com hàgim creat el model les diferencies entre les classes quedaran més ben reflectides en unes distancies o en altres.

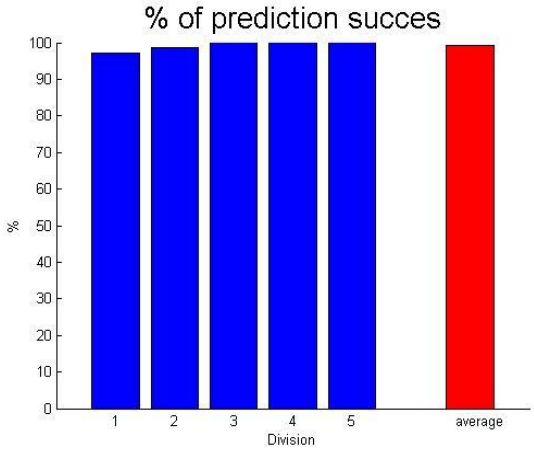
La resta de configuracions utilitzades les podem veure a la Figura 65.
A continuació mostrarem els resultats que retornen les execucions amb diferents distancies i diferents maneres de crear el model.

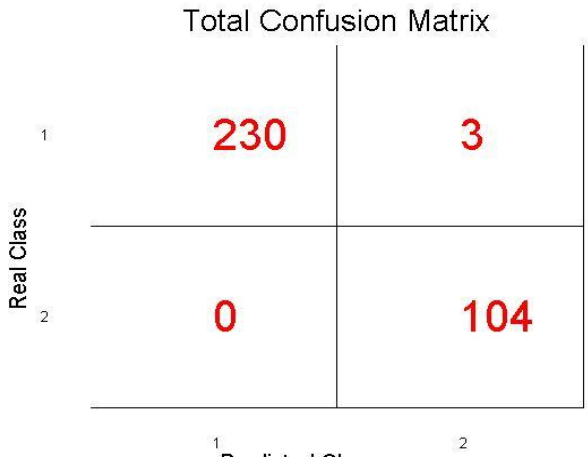
Distància sobre Q i model classe 1



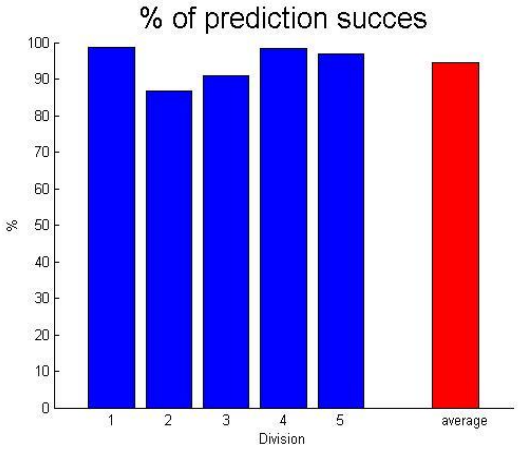


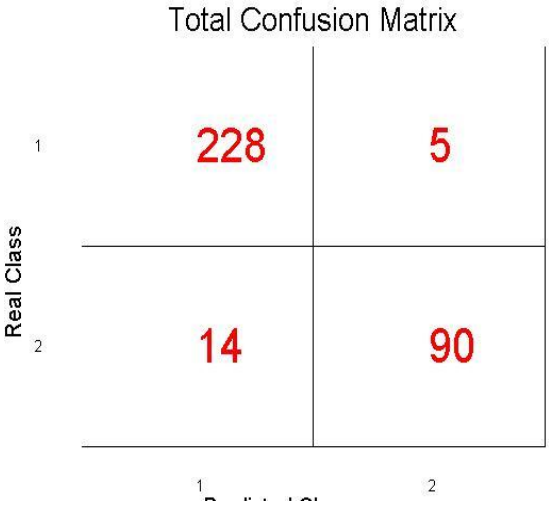
Distància Components principals i model classe 1



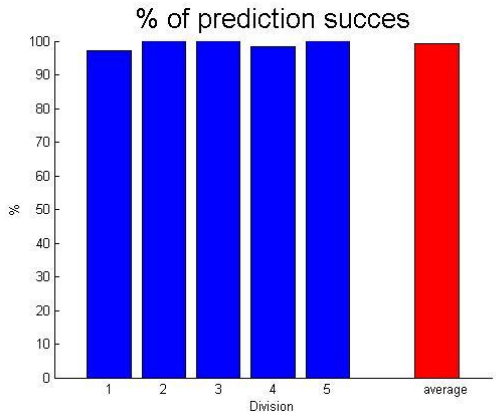


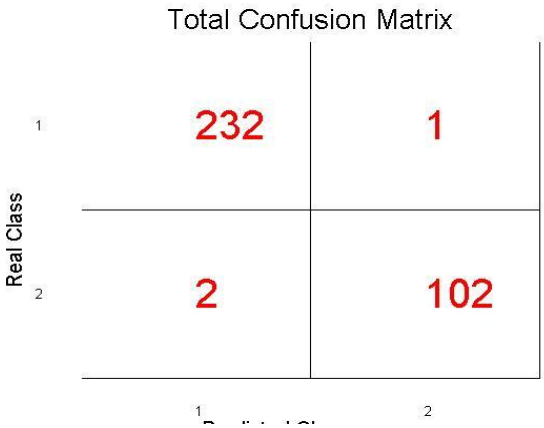
Distància sobre Q i model classe 2



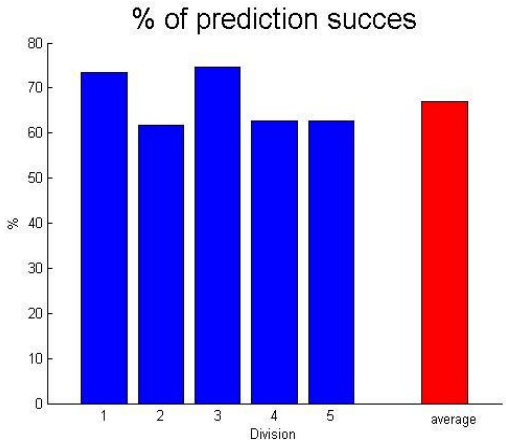


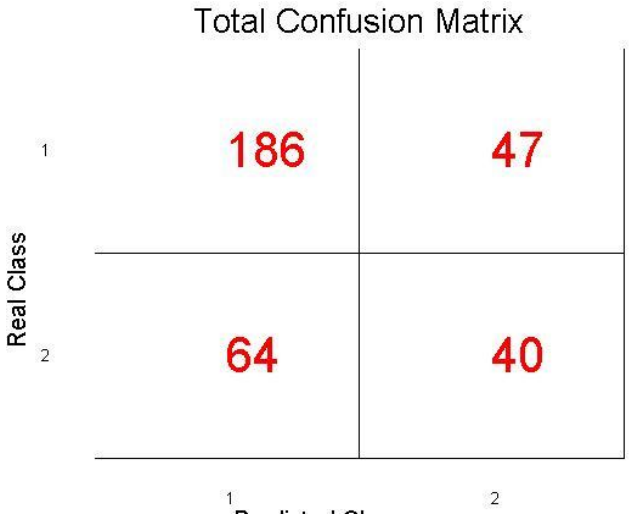
Distància Components principals i model classe 2



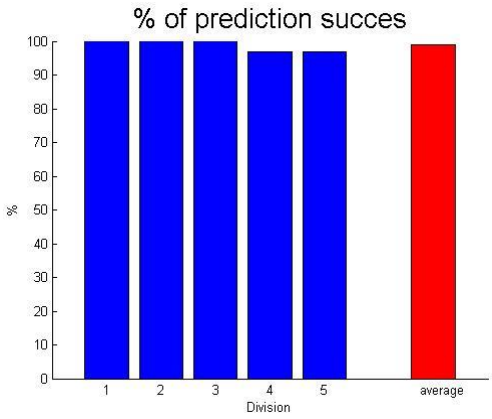


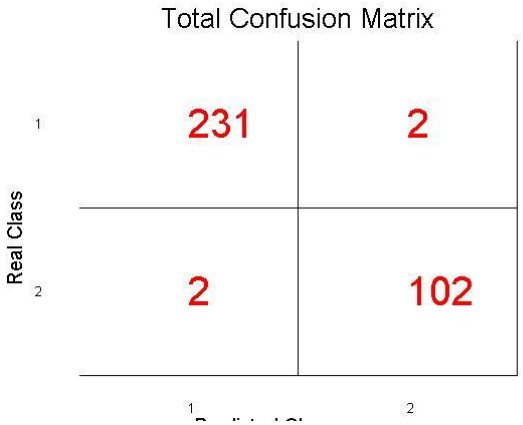
Distància sobre Q i model classe 1 i 3





Distància Components principals i model classe 1i3





Si ens fixem en els resultats podem veure fàcilment que la millor distància és sempre la que treballa sobre les components principals. Pel que fa a amb quina classe o classes fem el model no podem observar diferencies gaire notables ja que els 2 estats del procés són significativament diferents, a més es pot observar que els casos que fallen solen ser sempre els mateixos.

5.8 Creem un CBR utilitzant totes les dades que tenim

Un cop hem trobat una configuració que ens assegura un bon % de classificació el que farem és crear un CBR amb aquesta configuració i utilitzar-lo per assignar classes (flocular / granular) als elements que inicialment havíem classificat com a desconeguts.

La forma més ràpida de crear aquest objecte CBR és per comandes.

```
%Creem un nou objecte DATA
%aquest contindrà els casos necessaris per crear el model
Mod = dad.copy();

%treiem els casos que no volem al model
Mod.delete(Mod.getClass()==2);

%A Mod nomes queden els elements classe 1
%ja podem crear el model PCA, ho farem al mateix temps que creem el CBR
MYCBR=CBR(PCA(Mod),dad);%creem un CBR
```

Només quedarà configurar el CBR perquè treballi amb la configuració escollida i ja podrem començar a classificar nous casos o en el nostre cas els insegurs.

5.9 Utilitzem la funció tallar per obtenir un CBR especialitzat en l'amoni

Observat detingudament les contribucions de cada element es va poder observar que la fase anòxica aporta gran part de la informació sobre el canvi d'estat (És on s'observen les diferencies més significatives entre els 2 estats). El que farem a continuació és utilitzant la utilitat tallar crear un nou CBR focalitzat només en aquesta fase. Ja que això permetrà obtenir resultats sobre la granularitat del procés abans de que aquest hagi acabat, també al tractar menys dades serà en general més ràpid i ocuparà menys espai en memòria.

Per construir un CBR focalitzat en la fase anòxica utilitzarem al funció Tallar i Tallarem el CBR que tenim muntat dels apartats anteriors.
Si accedim a la pantalla Talla un cop escollit el CBR que volem tallar haurem d' escollir els instants de temps i les variables que volem que tingui el nou CBR.

En el nostre cas volem totes les variables i la fase anòxica que correspon a els instants de temps [12 : 144]. A la Figura 66 hi tenim com queda la pantalla amb la configuració desitjada.

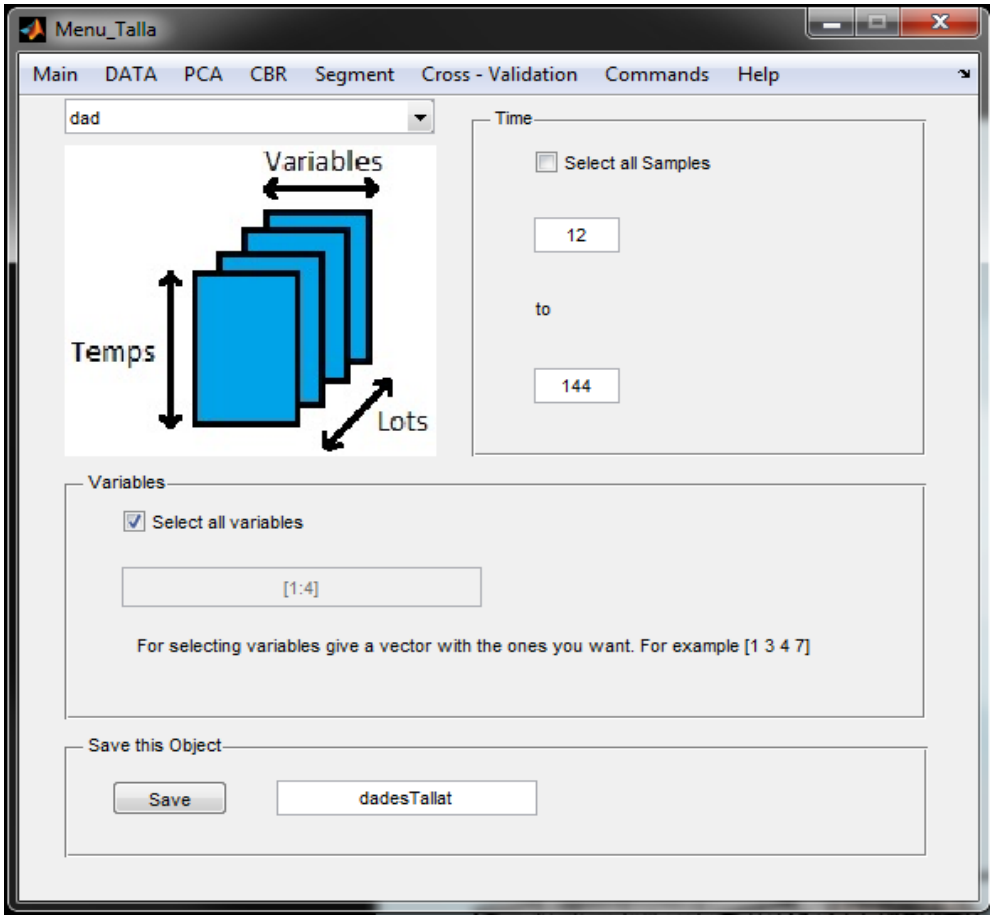


Figura 66

No ens hem d'oblidar de tallar també la base de casos de test, si no ho fem en intentar utilitzar els casos de test la interfície ens mostrarà un error dient que les mides no concorden.

5.10 Resultats

Els resultats obtinguts en fer un retrieve de tots els casos corresponents al període d'on no disposem de dades d'analítiques són els següents:

Informació de la configuració
Model = Casos {'24/09/2009' a '19/01/2010'} (corresponents a l'estat Granular)
Base de Casos= {'24/09/2009' a '20/01/2010'} + {'12/02/2010' a '23/03/2010'} (corresponen a tots el cassos Granulars i Flocular segurs)
distància = Components Principals
Without deleting outliers

Utilitzant totes les fases :

El reuse ha assignat les següents classes als casos que eren insegurs

{20/12/2009 - 10/01/2010} Class 1
{11/01/2010 - 19/01/2010} Class 3 excepte el dia {13/01/2010}
{19/01/2010 - 26/01/2010} Class 1
{26/01/2010 - 27/01/2010} Class 3
{27/01/2010 - 28/01/2010} Class 1
{29/01/2010 - 11/02/2010} Class 3

Exceptuant el període on el sensor de PH fallava observem que el canvi de granularitat es produeix clarament entre els dies 26/01/2010 i 28/01/2010. Els resultats també els trobem graficats a la Figura 67.

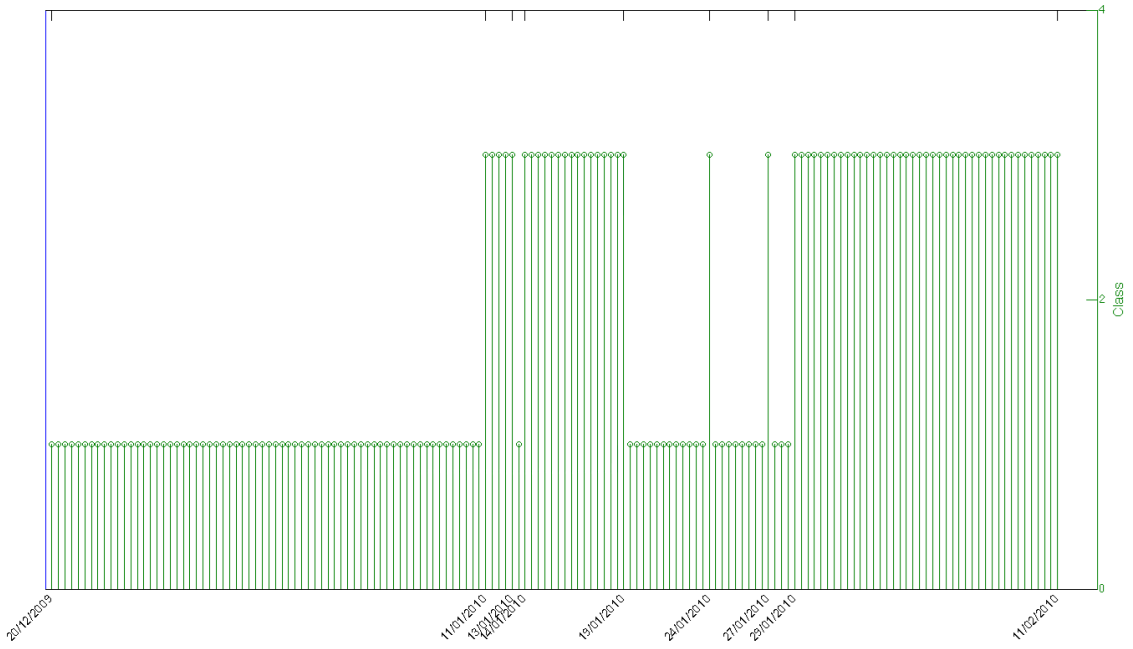


Figura 67

Utilitzant nomes la informació aportada per la fase anòxica

El reuse ha assignat les següents classes als casos que eren insegurs

{20/12/2009 - 10/01/2010} Class 1
{11/01/2010 - 19/01/2010} Class 3 excepte a 2 obs. {13/01/2010}
{19/01/2010 - 28/01/2010} Class 1 excepte a 1 obs. {27/01/2010}
{29/01/2010 - 11/02/2010} Class 3

Exceptuant el període on el sensor de PH fallava observem que el canvi de granularitat es produeix clarament entre els dies 27/01/2010 i 28/01/2010 . Els resultats son pràcticament els mateixos que utilitzant totes les fases.

5.11 Conclusions del cas d'aplicació

Observem que en el període que el sensor de PH falla ens equivoquem a l'hora de classificar (sabem que són floculars (classe 1) com a mínim fins al dia 12/1/2010) la resta del període es classifica aparent ment bé. A més a més podem observar que la transició la trobem entre els dies 26/01/2010 i 28/01/2010.

El fet que amb la fase anòxica es pugui determinar la granularitat és molt interessant perquè permet saber la granularitat abans que el procés hagi finalitzat i permetre que l'operari prengui les mesures adients.

6 Conclusions

En general els objectius marcats en començar el projecte han estat assolits. A continuació detallarem més concretament objectiu a objectiu.

-El primer objectiu era permetre la creació de models estadístics per al monitoratge de processos. Podem dir que ha estat assolit ja que a les llibreries desenvolupades permetem la creació de models estadístics mitjançant PCA.

-A PCA hi hem afegit les funcionalitats necessàries per què els usuaris puguin detectar la presencia de falles, entre les solucions implementades trobem les contribucions de Q i T2 entre d'altres. També es permet la classificació de manera automatitzada mitjançant l'algorisme CBR. Amb tot això hem obtingut un procediment de diagnosi de falles que era el nostre segon objectiu.

-En tercer lloc ens vam proposar la creació d'un conjunt de funcions per a la visualització dels diferents resultats, que facilitaran a l'operari la comprensió i identificació de l'estat del procés. S'han incorporat una gran quantitat de gràfiques i configuracions opcionals tant a l'algorisme PCA com a CBR per tal d'assolir l'objectiu.

-El quart objectiu s'ha aconseguit després de testejar la Toolbox amb diversos exemples i comparar resultats amb eines independents com la PLS Toolbox. Pel que fa al monitoratge de processos per lots s'han inclòs 2 modalitats de desdoblat i 4 d'escalat.

-El cinquè objectiu era la inclusió de funcionalitats que milloressin la qualitat del model MPCA. Com es comentava s'ha inclòs la possibilitat de detectar i eliminar d'outliers, subdivisió del procés mitjançant la funció talla. Podem dir doncs que

també ha estat assolit.

- El sisè i setè objectiu consistien a dotar a la llibreria de diversos formats per entrar les dades. Actualment disposem d'un format propi d'entrada i sortida de dades en format text , la possibilitat de carregar dades del workspace de Matlab fàcilment, la possibilitat d'exportar dades a weka, la possibilitat de guardar les dades en el format de la PLS Toolbox i funcions que permeten de manera externa l'accés des de Matlab a bases de dades MYSQL i parcejar els fitxers de les depuradores proporcionats per LEQUIA.

- Tal com es mencionava al setè objectiu s'ha dotat a la llibreria d'un seguit de interfícies per tal que a més de llibreria es pugui utilitzar el contingut com a aplicació capaç de fer el monitoratge de processos de manera autònoma.

Finalment es plantejava la possibilitat de crear un exemple com a validació de la Toolbox, implementar una aplicació capaç de detectar el nivell de granularitat d'un procés de depuració d'aigües. Aquesta aplicació la podem trobar inclosa en el CD, per manca de temps i degut a que era molt extens el temari al PFC només s'ha inclòs un exemple de detecció de granularitat fent servir les utilitats de la llibreria desenvolupada.

Per altra banda, cal també esmentar els punts on més tems hem hagut de dedicar durant la realització d'aquest PFC.

Optimització del codi

Una de les dificultats principals, si no la major, que ens hem trobat a l'hora de dur a terme el projecte ha estat aconseguir optimitzar el codi. Matlab és un llenguatge interpretat, això vol dir que si volem que l'execució del nostre codi sigui fluida cal optimitzar-lo molt i de manera manual.

Hem aplicat en tot moment els aspectes bàsics de l'optimització de codi en Matlab com són:

- Inicialitzar sempre que es pugui les variables i evitar canviar els tipus de les variables. (Per evitar que la màquina virtual estigui redimensionant constantment la variable)
- Utilitzar les operacions de matrius. Matlab disposa d'un ampli ventall de operacions matricials que són molt més eficients que qualsevol bucle que puguem fer nosaltres. Ja que algunes d'aquestes funcions en les últimes versions utilitzen CUDA per aprofitar el paral·lelisme.
- Evitar sempre que es pugui els bucles.
- Utilitzar l'estructura "switch-case" en comptes de "if" enniuats

En general cal recordar que el codi serà parcejat i executat tal qual està escrit línia a línia. Tenint en compte això es evident que cal vigilar de no fer càlculs innecessaris o redundants, treure els càlculs fora dels bucles sempre que sigui possible.

Un dels mètodes que més s'ha aconseguit optimitzat ha estat el DROP4. Aquest mètode és un doble bucle. Utilitzant operacions matricials i precalculant totes les distancies s'ha aconseguit "suprimir" un dels dos bucles, això ha suposat una millora notable en els temps d'execució (de 10 minuts de mitjana a 5 segons) però per contra el codi ha quedat molt poc intuïtiu i consumeix més memòria RAM.

Traducció de les funcions i interfície:

Inicialment el codi i les interfícies d'usuari estava tot en català. Per tal de permetre que el codi pogués ser utilitzat per gent d'altres països es va decidir en una de les reunions de traduir tots els missatges, noms de funcions i interfícies a l'Anglès.

Generació de la documentació:

Tot i que pot no semblar una dificultat, ha estat la part que més temps i esforç ha requerit de tot el projecte. Sobretot perquè és la primera vegada que havíem de generar un manual d'usuari i la documentació complerta d'unes llibreries. Per futurs projectes s'haurà de tenir en compte.

7 Treball futur

Hi ha diverses pantalles on s'ha trobat maneres de millorar-ne el rendiment. Però per dur-ho a terme es necessitava massa temps i no s'ha pogut desenvolupar a temps per incloure-ho al projecte. Aquestes millores haurien de permetre que la pantalla Veure Veïns, mostrés els resultats de manera molt més ràpida, i que l'exploració del workspace per buscar objectes fos molt més eficient.

Pel que fa al retreave i reuse s'han trobat unes possibles millores en la presentació dels resultats per fer més fàcil la interacció amb l'usuari a la pantalla menu_CBR.

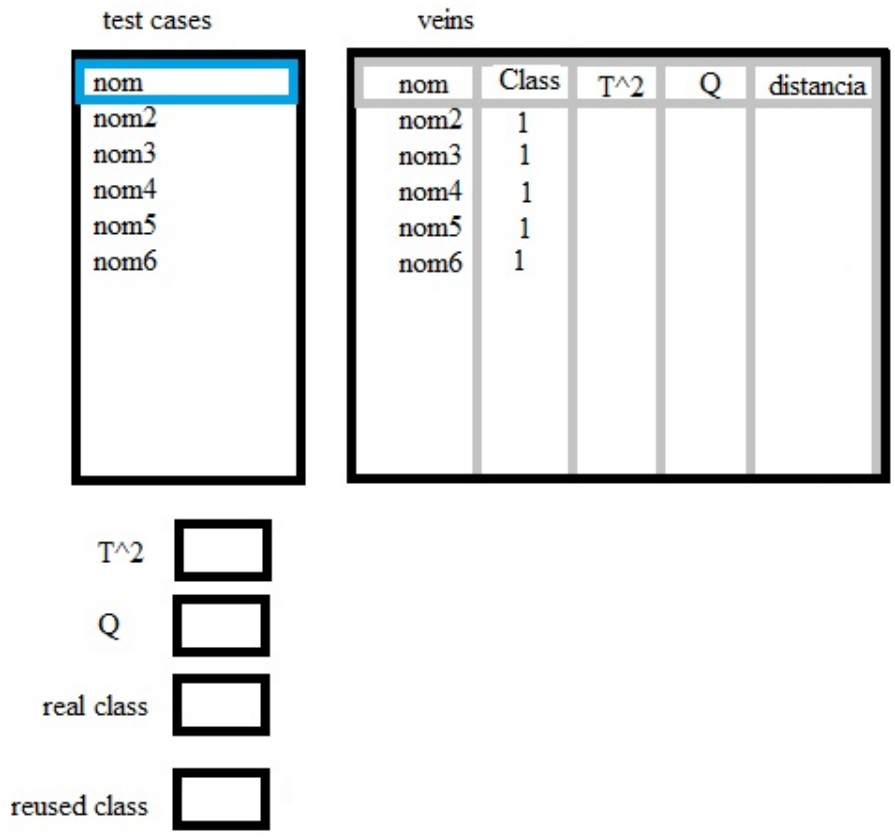


Figura 68

Juntament amb aquestes millores de presentació de resultats es podria afegir la possibilitat de mostrar gràfics per comparar les variables del cas de test i els seus veïns per poder veure de manera ràpida les característiques dels casos associats (croquis de la Figura 68).

També es podria permetre a l'usuari afegir i treure casos de la Base de casos manualment (sense utilitzar les funcions d'aprenentatge característiques de CBR) des de la interfície d'usuari, actualment només es pot fer mitjançant comandes.

Del cas d'aplicació també se'n podria extreure una aplicació per als operaris de les plantes depuradores d'aigües. Aquesta aplicació donat un model pre-configurat per la configuració especifica de cada planta, nomes s'hi permetria a l'operari testejar i visualitzar casos carregant-los directament des dels fitxers que retorna el procés. Una possible aparença per l'aplicació seria el que es mostra a la Figura 69. Una aplicació d'aquest tipus seria una eina útil ter saber l'estat de granularitat del procés.

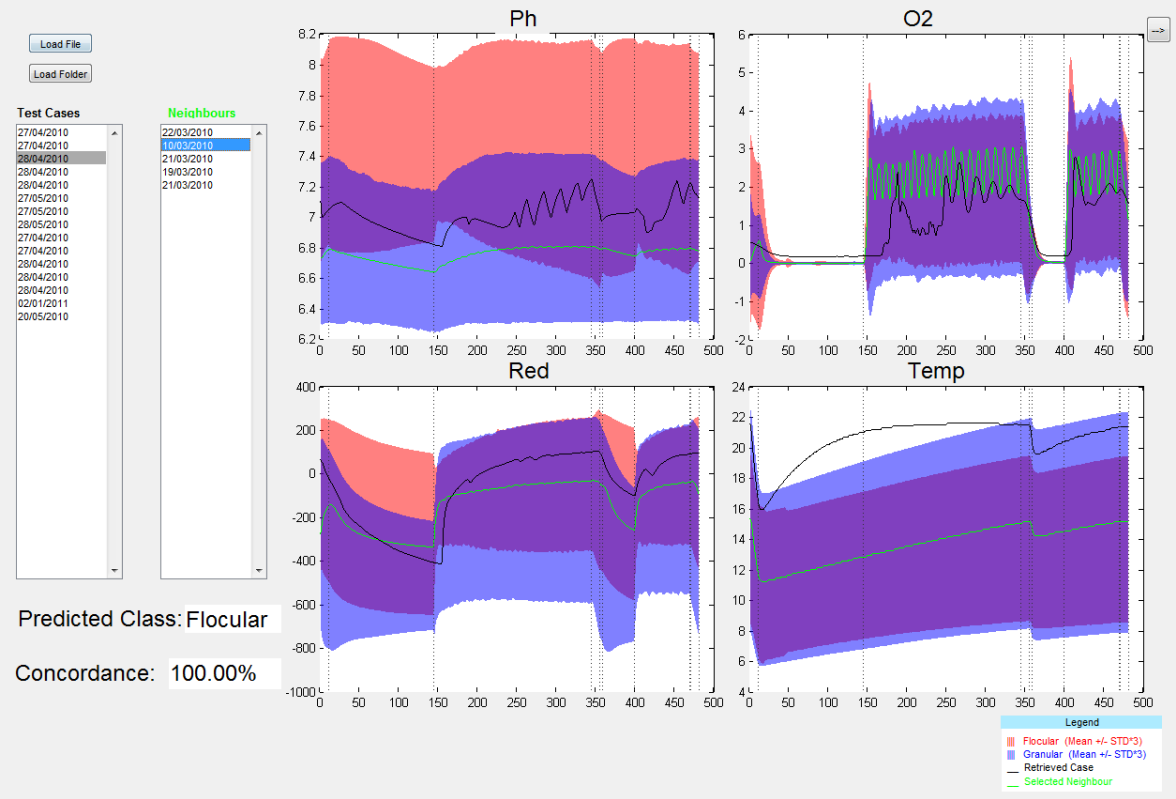

Load File
Load Folder
Test Cases
Neighbours
27/04/2010
27/04/2010
28/04/2010
28/04/2010
28/04/2010
27/05/2010
27/05/2010
28/05/2010
27/04/2010
27/04/2010
28/04/2010
28/04/2010
28/04/2010
02/01/2011
20/05/2010
22/03/2010
10/03/2010
21/03/2010
19/03/2010
21/03/2010
Predicted Class: Flocular
Concordance: 100.00%
Ph
O2
Red
Temp
Legend
Flocular (Mean +/- STD*3)
Granular (Mean +/- STD*3)
Retrieved Case
Selected Neighbour

Figura 69

8 Bibliografia

X. Berjaga. *Case-based diagnosis in the principal components space. from definition to applications.* Master's thesis, Universitat de Girona, 2008.

X. Berjaga, M. Coma, J. Meléndez, S. Puig, J. Colprim and J. Colomer, "Granularity determination of activated sludge through on-line profiles by means of case-based reasoning," in Bioprocess and Biosystems Engineering (under review), 2012

A. Aamodt i E. Plaza. *Case-based reasoning: foundational issues,methodological variations, and system approaches.* 1994

D. Aha, D. Kibler, i M. Albert. *Instance-based learning algorithms. Machine Learning,* 1991.

X. Berjaga i A. Pallarés, *A Framework for Case-Based Diagnosis of Batch Processes in the Principal Components Space*.